REPORT T-17 JUNE, 1975

CSL COORDINATED SCIENCE LABORATORY

LINSYS CONVERSATIONAL SOFTWARE FOR ANALYSIS AND DESIGN OF LINEAR SYSTEMS

STANOJE BINGULAC

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

LINSYS

CONVERSATIONAL SOFTWARE FOR ANALYSIS AND DESIGN OF LINEAR SYSTEMS

CONTENTS

INTRODUCTION

This manual describes conversational software LINSYS intended for analysis and design of linear systems. The use of LINSYS does not require knowledge of computer programming. LINSYS is written in FORTRAN-IV and is executable on the DEC-10 computer having Advance Graphic Software AG-11 [1] and IBM Scientific Subroutine Package SSP-Version III [2]. Modular structure of LINSYS permits easy modifications and extensions. Subroutines used within LINSYS may be used independently or combined for solving a specific problem in the area of linear system theory.

Basically, LINSYS consists of a main program and three packages MRIC, MPPL and MANAL. MRIC solves the algebraic Riccati equation for optimal control problem and Kalman filter design [3]. MPPL is used for pole placement and observer design. In contrast to MRIC and MPPL which are design-oriented, package MANAL is analysis-oriented. Among its options are controllability and observability tests, transformation to the Jordan form, state transition matrix calculation, etc.

All packages plot eigenvalues locations and time-domain responses to initial conditions and polynomial input signals. Full capabilities of LINSYS are attained by its execution from the TEKTRONIX computer terminal supported by the AG-II graphic software. However, LINSYS may also be executed from other computer CRT or TTY terminals. Input data are entered either from a computer terminal or from a disk data file. All inputs from the terminal are format-free. Numerical results are displayed on the terminal, printed on a line printer or stored on a disk data file. These options are selected in a conversational mode during LINSYS execution.

Graphical results are displayed on the TEKTRONIX terminal. Also, an integral part of the software LINSYS is a package EDITD used for editing the disk data file. By means of the EDITD it is possible to enter new data, delete existing records, or change particular numerical values within an existing record.

This document is a preliminary report on the author's work during his visit at the Coordinated Science Laboratory, University of Illinois, February 1975 - June 1975. In such a short time it was impossible to test all the parts of the software. The author apologizes for any difficulties which may arise in applications and would appreciate suggestions and comments.

ORGANIZATION OF THE MANUAL

In this manual, emphasis will be placed on the conversational program elements; i.e. those which interact directly with the user. For other subroutines used exclusively in calculations, only the calling sequences are explained. Complete write-ups of all developed subroutines are also included. Users experienced in FORTRAN programming can combine available subroutine for solving control problems not covered by LINSYS. Illustrative examples given at the end of the manual also clarify the use of LINSYS.

ACKNOWLEDGMENTS

The author wishes to thank his colleagues and the many students of the Control Systems Group, Coordinated Science Laboratory, who have contributed to this project. Special thanks are due to Professors P. V. Kokotovic, J. B. Cruz, Jr., and W. R. Perkins for their continuous interest, encouragement and suggestions. Mr. T. W. Burtnett has helped the author to avoid some difficulties related to DEC-10 FORTRAN Compilers and contributed to the design of the EDITD package. Mr. W. Labiak tested many of the included subroutines. Mr. D. Daly has used a part of the software and gave many useful suggestions. The assistance of Mr. D. Barbour in suggesting certain corrections in the early version of the manuscript is appreciated.

I. CONVERSATIONAL ROUTINES

1. LINSYS DESCRIPTION

A general flowchart of LINSYS is given in Fig. 1, showing how the control is transferred to either of the three packages. LINSYS execution starts by typing on TTY the message:

```
IN CASE OF USING DISK DATA FILE IT IS NECESSARY
TO ASSIGN DSK:2.  IT MAY BE DONE BY TYPING:
⟨CTRL⟩C;  ⟨ASS DSK:2⟩,⟨CR⟩ AND ⟨CONT⟩,⟨CR⟩
```

informing the user that, if disk data file is to be used, then it is necessary to assign disk #2. The disk may be assigned either before or during the execution.* Functions of particular blocks on the flowchart are as follows.

Block #1. Plots on the terminal screen system block diagram, Fig. 2, which identifies some of the matrices and variables used within LINSYS. In order to permit LINSYS execution on terminals not supported by the AG-II software, first the following message is typed on TTY

```
WANT SYSTEM BLOCK DIAGRAM - Y OR N
```

Reply 'Y' transfers the control to the subroutine BLDG which plots the block diagram. Reply 'N' bypasses the block diagram plotting.

---

*Automatic disk assignment may be done by including into user's area the following file:

```
SWITCH.INI
LOGIN/ASSIGN:DSK:2/ASSIGN:DSK:3/ASSIGN:DSK:4
```

Disks #3 and #4 are used for editing the disk data file.

Block #2. Within this block the user has the opportunity to transfer the control to either the package MRIC, MPPL, MANAL or to stop execution. This block is implemented by the subroutine IFFM. The subroutine types on TTY the message

```
TYPE ⟨R⟩,⟨P⟩,⟨A⟩,⟨S⟩ OR HELP
```

expecting as a reply either of the characters:

'R', 'P', 'A', 'S', 'H' (or HELP).

Replies 'R', 'P' or 'A' transfer the control to packages MRIC, MPPL or MANAL, respectively, while 'S' stops the execution. After the reply 'H' (or HELP) the following message is typed on TTY

```
TO RUN RICCATI PACKAGE;
       POLE PLACEMENT/OBSERVER PACKAGE;
       ANALYSIS PACKAGE; OR TO
       STOP
TYPE ⟨R⟩,⟨P⟩,⟨A⟩,⟨S⟩ OR HELP
```

which should assist inexperienced users in selecting a desired option.

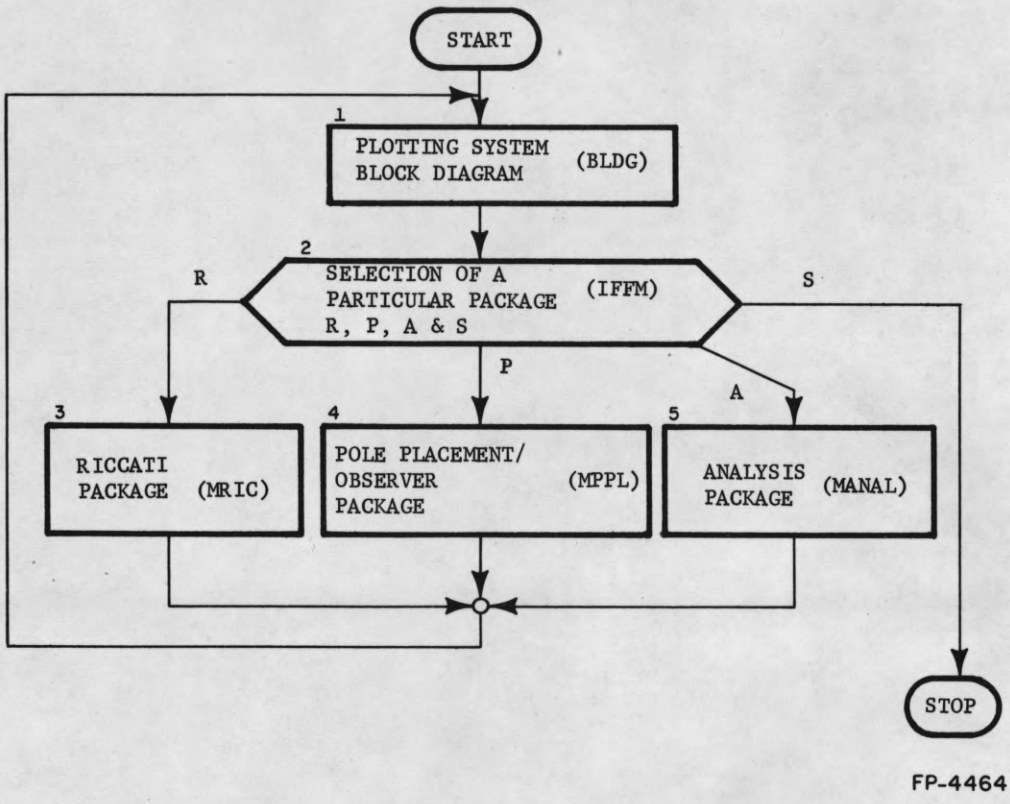

START
1
PLOTTING SYSTEM BLOCK DIAGRAM (BLDG)
2
SELECTION OF A PARTICULAR PACKAGE R, P, A & S (IFFM)
R
P
A
S
3
RICCATI PACKAGE (MRIC)
4
POLE PLACEMENT/ OBSERVER PACKAGE (MPPL)
5
ANALYSIS PACKAGE (MANAL)
STOP
FP-4464

Figure 1

2. BLOCK DIAGRAM DESCRIPTION

The linear systems treated by LINSYS are represented by

$$\begin{aligned} \dot{x}(t) &= A\,x(t) + B\,u(t) + w(t); \quad x(0) = x^{o} \\ y(t) &= C\,x(t) \end{aligned} \tag{2.1}$$

where $x(t) \in R^{n}$; $y(t) \in R^{k}$; $u(t) \in R^{m}$; $w(t) \in R^{n}$ are state, output, control and disturbance vectors, respectively. In the optimal control and pole placement, packages MRIC and MPPL, it is assumed that $k = n$ and $C = I$, and that $u(t)$ is given by

$$u(t) = F\,x(t), \tag{2.2a}$$

while in MANAL $u(t)$ is given by

$$u(t) = F\,y(t). \tag{2.2b}$$

In the state observer, package MPPL, the control vector $u(t)$ is given by

$$u(t) = F\,\hat{x}(t) \tag{2.2c}$$

where $\hat{x}(t)$ is the estimated state. In (2.1) and (2.2)

$A = (n \times n)$ system matrix

$B = (n \times m)$ input matrix

$C = (k \times n)$ output matrix

$F = (m \times k)$ feedback gain matrix.

The disturbance vector $w(t)$ is assumed to be in the polynomial form i.e.

$$w(t) = \sum_{i=1}^{n_d} e^{i}\, t^{i-1} = E\,z(t)$$

where $(n \times n_d)$ disturbance matrix E has a form

$$E = |e^1 \vdots e^2 \vdots \cdots \vdots e^{n_d}|$$

while elements of the n_d dimensional column vector $z(t)$

$$z(t) = | \; 1 \quad t \quad t^2 \ldots t^{n_d-1} \; |^T$$

are step, ramp etc.

The block diagram plotted on the terminal screen is shown in an example in Sec. 16.

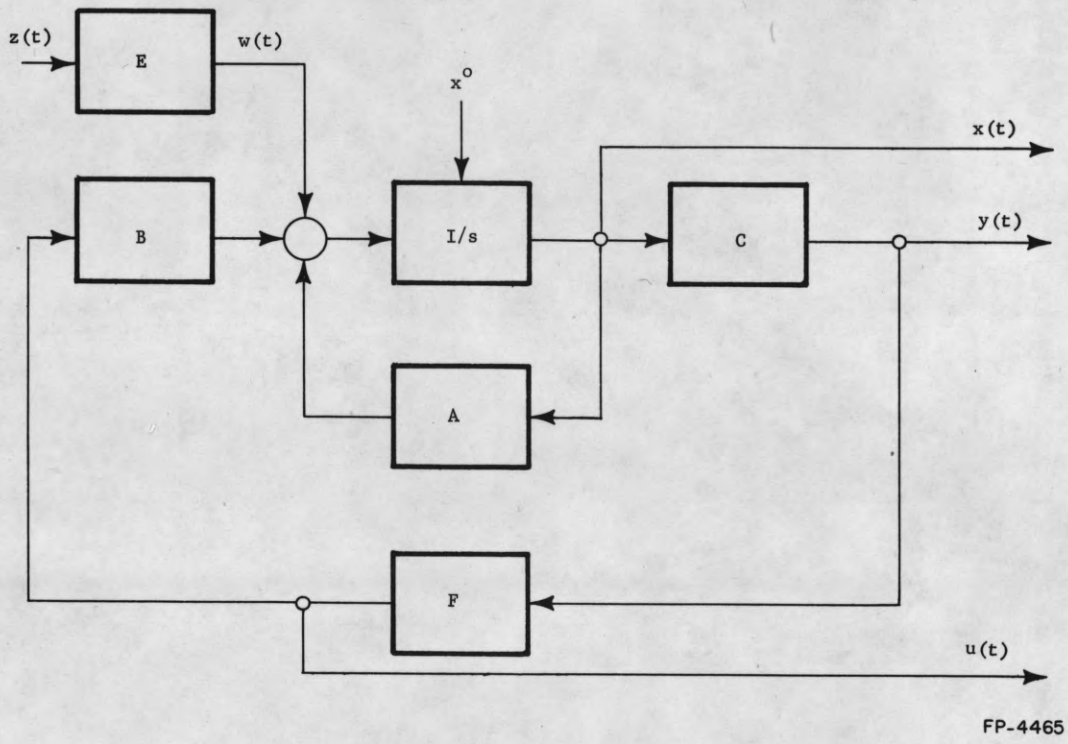


Figure 2

3. PACKAGE MRIC

BACKGROUND

The package MRIC calculates the feedback gain matrix F, given by

$$F = -R^{-1} B^T K$$

where $(n \times n)$ Riccati matrix gain K satisfies the Riccati algebraic equation

$$I(K) = KA + A^T K - KSK + C^{*T} C^* = 0; \quad S = BR^{-1} B^T. \tag{3.1}$$

Matrices A, B and F are defined by (2.1) and (2.2a). R and C* are $(m \times m)$ and $(k \times n)$ matrices, respectively defining the performance index

$$J(K) = \int_0^\infty [x^T(t) C^{*T} C^* x(t) + u^T(t) R u(t)] dt; \quad R > 0,$$

to be minimized.

The Riccati equation (3.1) is solved by the Newton-Lyapunov iterative method [5] with the stopping condition

$$\|I(K)\| / \|K\| \leq e = 10^{i_e} \ll 1; \quad i_e < 0.$$

The iterative process stops also if the number of iterations exceeds the number i_t, specified in advance. The package checks the controllability (stabilizability) and observability (detectability) of the pairs $\{A,B\}$ and $\{A,C^*\}$, respectively. For these tests, a sufficiently small number e_c

$$e_c = 10^{i_c} \ll 1 ; \quad i_c < 0$$

is required. Also, locations of closed-loop system eigenvalues and time responses x(t) and u(t) are plotted on the terminal screen.

FLOWCHART DESCRIPTION

The execution of the package MRIC starts by typing on TTY the message

YOU ARE NOW ENTERING INTO THE RICCATI PACKAGE
ENTER WITH N,M,K,IT,IE,IC

Block #1. Integers

N, M, K, IT, IE and IC

corresponding to previously defined

n, m, k, i_t, i_e and i_c

are entered via terminal keyboard TTY. Integer input is format-free. As delimiter the comma ',' is used. Default values for M, K, IT, IE and IC are

M = 1

K = N

IT = 10

IE = -6

IC = -2.

Block #2. The message

AVAILABLE NAMES ARE ⟨A ⟩⟨B ⟩⟨C ⟩⟨R ⟩⟨K ⟩
TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP

is typed on TTY. Matrices

A, B, C^*, R and K^o

are entered via TTY or disk data file (DSK). K^o represents the initial guess for the Riccati gain matrix K. If no better K^o is known, $K^o = 0$ may be used [6]. Individual element, row, column or the main diagonal of already entered matrices can be changed via TTY without retyping the whole matrix. Matrices entered from TTY are typed format-free, row-wise. As delimiter comma ',' is used. For details refer to the subroutines INPDAT, INPTTY, INPDSK and particularly to the subroutine IOMT. The subroutine TYPDIM types on TTY matrix names and their dimensions. Examples of this message is

```
  A       B       C       R       K
< 5, 5> < 5, 3> < 4, 5> < 3, 3> < 5, 5>
```

Block #3. The following message

```
AVAILABLE NAMES ARE <A  ><B  ><C  ><R  ><K  >
 TYPE <T>,<L>,<D>,<N> OR HELP
```

is typed on TTY. This block permits either typing on TTY, printing on LPT or storing on DSK some of the entered matrices. See subroutine OUTDAT.

Block #4. Controllability (stabilizability) and observability (detectability) of the specified system is checked. If the system is both controllable and observable the following message

```
SYSTEM CONTROLLABLE
SYSTEM OBSERVABLE
```

is typed on TTY. See subroutine RANKR. Otherwise, the subroutine indicates system mode which is either not controllable or not observable. In the example of not detectable and not controllable but stabilizable system, the

following information is typed on the TTY:

```
MODE  -0.200E+01  +/- J   0.000E+00  NOT CONTROLLABLE
MODE   0.300E+01  +/- J   0.000E+00  NOT OBSERVABLE
```

Block #5. The subroutine IFFM executed within this block types on TTY the message

```
TYPE ⟨I⟩,⟨E⟩,⟨C⟩ OR HELP
```

As indicated in the flowchart, typing either 'I' or 'E' transfers control to the block #1 or #2, allowing in this way the introduction of additional changes into either integer values or matrix elements. Typing the reply 'C' transfers the control to block #6. In the case of typing the reply 'H' the following message is typed on TTY

```
TO CHANGE INTEGERS (MATRIX DIMENSIONS , ETC.);
          ELEMENTS OF MATRICES; OR TO
          CONTINUE
TYPE ⟨I⟩,⟨E⟩,⟨C⟩ OR HELP
```

which facilitates selection of the desired option.

Block #6. Solution of the algebraic Riccati equation (3.1), calculation of the closed-loop system matrix $A_f = A - S\,K$ and calculation of the regulator gain matrix $F = -R^{-1}B^T K$ are done by the subroutine RIC. Upon completing the solution, RIC types on TTY the values of a quantity $e_{rr} = \|I(K)\|/\|K\|$ obtained in each iteration. If the pair $\{A,B\}$ is not stabilizable, then a solution K satisfying (3.1) does not exist. In this case the subroutine types on TTY the message

```
SYSTEM NOT STABILIZABLE
AVAILABLE NAMES ARE ⟨A  ⟩⟨B  ⟩⟨C  ⟩⟨R  ⟩⟨K  ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

and control is transferred to block #2.

Block #7. A part of the s-plane containing all or only dominant eigenvalues of the closed-loop system are plotted on the terminal screen. The selection of the part of the s-plane to be plotted is done in a conversational mode within the subroutine EGVP. The subroutine EGVP starts execution by typing on the TTY the message:

```
DO YOU WANT LOCATIONS OF EIG.VAL.-Y OR N
```

The reply 'N' bypasses the eigenvalues plotting.

Block #8. This block is implemented by two calls of the subroutine OUTDAT, which type on TTY the following messages

```
AVAILABLE NAMES ARE ⟨A  ⟩⟨B  ⟩⟨C  ⟩⟨R  ⟩⟨K  ⟩⟨Q  ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

and

```
AVAILABLE NAMES ARE ⟨RR ⟩⟨RI ⟩⟨FR ⟩⟨AF ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

This block permits either typing on TTY, printing on LPT or storing on DSK some of the matrices among

$$A,\ B,\ C,\ R^{-1},\ K,\ Q,\ RR,\ RI,\ F \text{ and } AF$$

where RR and RI are real and imaginary parts of the eigenvalues of the closed-loop system matrix AF. See subroutine OUTDAT.

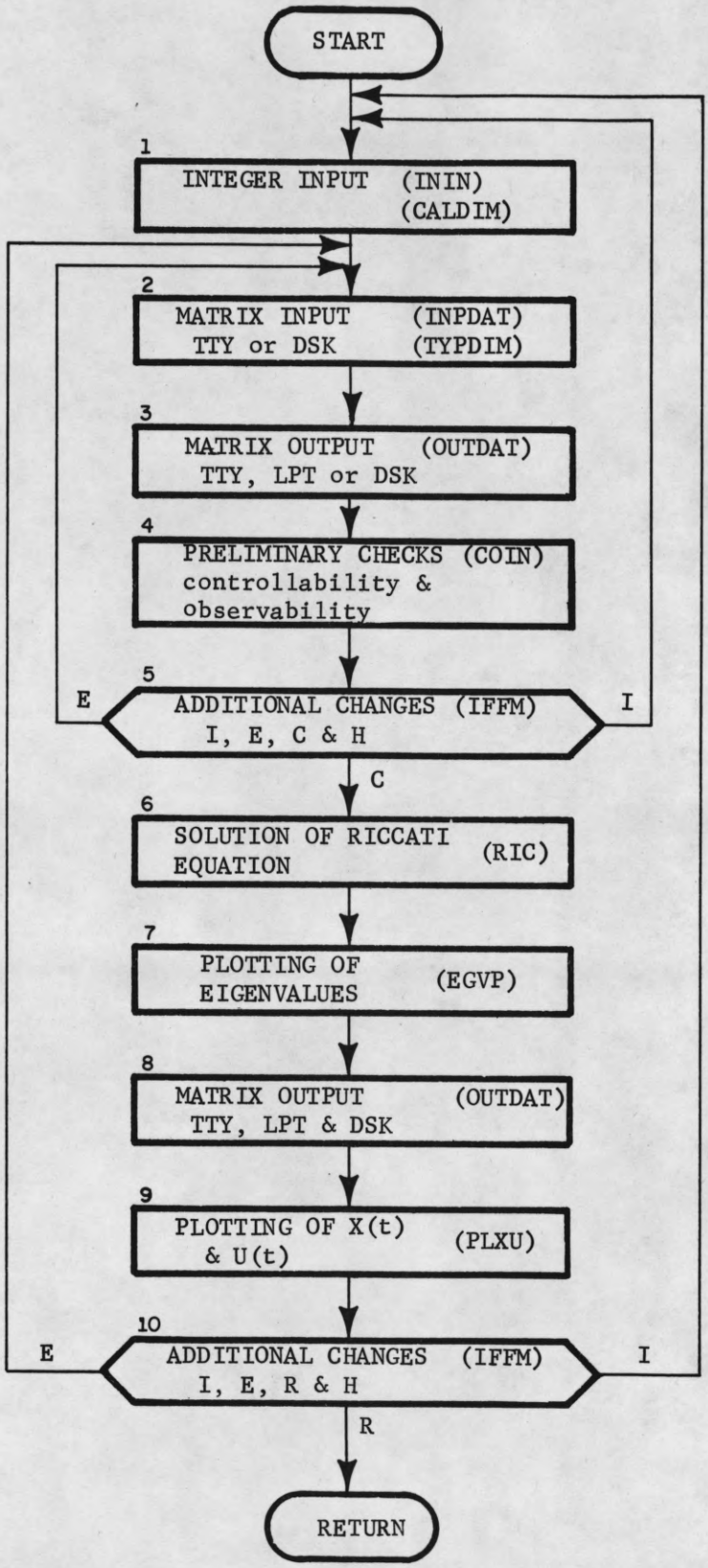
Block #9. Plotting on the terminal screen elements of the state and control vectors x(t), and u(t), for $t \in [0,T]$. Selection of indices of x(t) and u(t) to be plotted as well as introduction of numeric values for n_d, elements of the $(n \times n_d)$ matrix E, initial condition vector x^o and the time interval T is done in the conversational mode within the subroutine PLXU. See block diagram on Fig. 2.

Block #10. The subroutine IFFM types on TTY the message

```
TYPE ⟨I⟩,⟨E⟩,⟨R⟩ OF HELP
```

Typing either 'I', 'E' or 'R' transfers the control to the block #1, #2 of the package MRIC, or returns to the block #1 of LINSYS allowing execution of another package within the LINSYS. In the case of the reply 'H' the following message is typed on TTY

```
TO CHANGE INTEGERS (MATRIX DIMENSIONS, ETC.);
        ELEMENTS OF MATRICES OR TO
        RETURN TO LINSYS
TYPE ⟨I⟩,⟨E⟩,⟨R⟩ OF HELP
```



FP-4466

Figure 3

4. PACKAGE MPPL

BACKGROUND

The package MPPL is used for pole placement by state feedback and observer design. It calculates the feedback gain matrix F necessary to attain a closed-loop system matrix

$$A_f = A + BF \tag{4.1}$$

having prespecifed eigenvalues $\lambda_i = \delta_i + j\omega_i$. Real and imaginary parts of λ_i are specified by n dimensional arrays RP and IP, respectively. If rank $|C| < n$, MPPL also calculates $(k \times n)$ observer gain matrix K necessary to attain a closed-loop observer matrix $A_{of} = A + KC$ having prespecified eigenvalues $\lambda_{oi} = \delta_{oi} + j\omega_{oi}$. Real and imaginary parts of λ_{oi} are specified by n dimensional arrays RO and IO, respectively. The calculation of both matrices F and K is done by the subroutine PPL. The subroutine PPL requires sufficiently small number e_c defined by

$$e_c = 10^{i_c} \ll 1; \quad i_c < 0.$$

For detailed description of the algorithm used for calculation of matrices F and K see subroutine PPL.

FLOWCHART DESCRIPTION

The first four blocks are basically the same as in the package MRIC.

The execution of the package MPPL starts by typing on TTY the message

YOU ARE NOW ENTERING INTO THE POLE PLACEMENT' /OBSERVER PACKAGE
ENTER WITH N,M,K, IC

Block #1. Integers N, M, K and IC corresponding to previously defined n, m, k and i_c are entered via TTY. Default values for M, K and IC are

$$M = 1$$

$$K = 1$$

$$IC = -2.$$

Block #2. The message

```
AVAILABLE NAMES ARE ⟨A  ⟩⟨B  ⟩⟨RP  ⟩⟨IP  ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Matrices A, B and arrays RP and IP are entered either from TTY or DSK. Names and dimensions of entered matrices are typed on TTY by the subroutine TYPDIM. An example of this message is

```
  A       B       RP      IP
⟨ 4, 4⟩ ⟨ 4, 2⟩ ⟨ 1, 4⟩ ⟨ 1, 4⟩
```

Block #3. The message

```
AVAILABLE NAMES ARE ⟨A  ⟩⟨B  ⟩⟨RP  ⟩⟨IP  ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Matrices A, B and arrays RP and IP are either typed on TTY, printed on LPT or stored on DSK.

Block #4. Same as block #5 in MRIC. The message

```
TYPE ⟨I⟩,⟨E⟩,⟨C⟩ OR HELP
```

is typed on TTY. Replies 'I', 'E' and 'C' transfer controls to blocks #1, #2 or #5, respectively.

Block #5. Subroutine PPL performs pole placement algorithm, see subroutine PPL. It calculates matrices F, A_f and product BF. If the pair {A,B} is uncontrollable or rank|B| < m, PPL types either of the messages

```
SYSTEM IS NOT CONTROLLABLE/OBSERVABLE
AVAILABLE NAMES ARE ⟨A ⟩⟨B ⟩⟨RP ⟩⟨IP ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

or

```
MATRIX B/C IS NOT OF FULL RANK
AVAILABLE NAMES ARE ⟨A ⟩⟨B ⟩⟨RP ⟩⟨IP ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

and the control is transferred to the block #2. If {A,B} is controllable and rank|B| = m the subroutine types on TTY values of controllability indices and ordered controllability indices. An example of this message is

```
CONTROLLABILITY/OBSERVABILITY INDICES  3  1
ORDERED CONTROLLABILITY/OBSERVABILITY INDICES  1  3
```

Ordered controllability indices are used for calculation of the matrix F, see the subroutine PPL.

Block #6. The message

```
AVAILABLE NAMES ARE ⟨A ⟩⟨B ⟩⟨RP ⟩⟨IP ⟩⟨F ⟩⟨AF ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Some of the matrices and arrays among

A, B, RP, IP, F and AF

are either typed on TTY, printed on LPT or stored on DSK.

Block #7. The message

DO YOU WANT STATE OBSERVER - Y OR N

is typed on TTY. The reply 'Y' transfers control to the block #8 permitting observer design. Reply 'N' bypasses the observer design and transfers control to the block #13.

Block #8. The message

```
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Output matrix C and arrays RO and IO specifying the desired observer eigenvalues are entered. Names and dimensions of C, RO and IO are typed on TTY. An example of this message is

```
  C       RO      IO
⟨ 3, 4⟩ ⟨ 1, 4⟩ ⟨ 1, 4⟩
```

Block #9. The message

```
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. C, RO and IO are either typed on TTY, printed on LPT or stored on DSK.

Block #10. Same as block #4. The message

TYPE ⟨I⟩,⟨E⟩,⟨C⟩ OR HELP

is typed on TTY. Replies 'I' and 'E' transfer control to the blocks #1 and #8, respectively, while the reply 'C' transfers control to the block #11.

Block #11. The subroutine PPL performs observer design. It calculates matrices K, A_{of} and the product KC. If the pair is unobservable, or rank$|C| < k$ PPL types on TTY the messages

```
SYSTEM IS NOT CONTROLLABLE/OBSERVABLE
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

or

```
MATRIX B/C IS NOT OF FULL RANK
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

respectively, and the control is transferred to block #8. If {A,C} is observable and rank $|C| = k$, the subroutine types on TTY values of observability indices and ordered observability indices, required for calculation of the matrix K. An example of this message is

```
CONTROLLABILITY/OBSERVABILITY INDICES  2  1  1
ORDERED CONTROLLABILITY/OBSERVABILITY INDICES 1  1  2
```

The subroutine JOIN forms a composite matrix

$$A_t = \left| \begin{array}{c|c} A+BF & -BF \\ \hline 0 & A+KC \end{array} \right| \qquad (4.2)$$

corresponding to a system composed of both the considered system and its observer [4], i.e.

$$\left|\begin{array}{c}\dot{x}(t)\\ \hline \dot{e}(t)\end{array}\right| = A_t \left|\begin{array}{c}x(t)\\ \hline e(t)\end{array}\right| + E\, z(t); \quad \left|\begin{array}{c}x(0)\\ \hline e(0)\end{array}\right| = \left|\begin{array}{c}x^o\\ \hline e^o\end{array}\right| \tag{4.3}$$

n dimensional vector e(t) represents the error signal $e(t) = x(t) - \hat{x}(t)$, $\hat{x}(t)$ being observed state generated by the observer. In this case the control vector u(t) is given by

$$u(t) = F\,\hat{x}(t) = \left|F \vdots -F\right| \left|\begin{array}{c}x(t)\\ \hline e(t)\end{array}\right| .$$

Block #12. This block is implemented by two calls of the subroutine OUTDAT, which type on TTY the following messages

```
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩⟨K  ⟩⟨AOF⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

and

```
AVAILABLE NAMES ARE ⟨AT ⟩⟨RT ⟩⟨IT ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

This block permits either typing on TTY, printing on LPT or storing on DSK some of the matrices and vectors among

C, RO, IO, K, AOF, AT, RT, and IT

where RT and IT are 2n dimensional vectors containing real and imaginary parts of the eigenvalues of the composite matrix AT, given by (4.2).

Block #13. Depending on the path how this block has been reached, subroutines EGVP and PLXU plot eigenvalue locations and time domain responses x(t) and u(t) of either the closed-loop system (2.1), (2.2a) or the composite system (4.3).

Block #14. The message

TYPE ⟨I⟩,⟨P⟩,⟨O⟩,⟨R⟩ OR HELP

is typed on TTY. Similarly as in other blocks implemented by the subroutine IFFM, this block permits either introduction of additional changes into integer values (reply 'I'), data specifying pole placement (reply 'P') or data specifying observer design (reply 'O'). Reply 'R' returns control to the block #1 of LINSYS which permits selection of another package. In the case of the reply 'H' (or HELP) the following message is typed on TTY

```
TO CHANGE INTEGERS; DATA FOR
          POLE PLACEMENT; DATA FOR
          OBSERVER DESIGN; OR TO
          RETURN TO LINSYS
TYPE ⟨I⟩,⟨P⟩,⟨O⟩,⟨R⟩ OR HELP
```

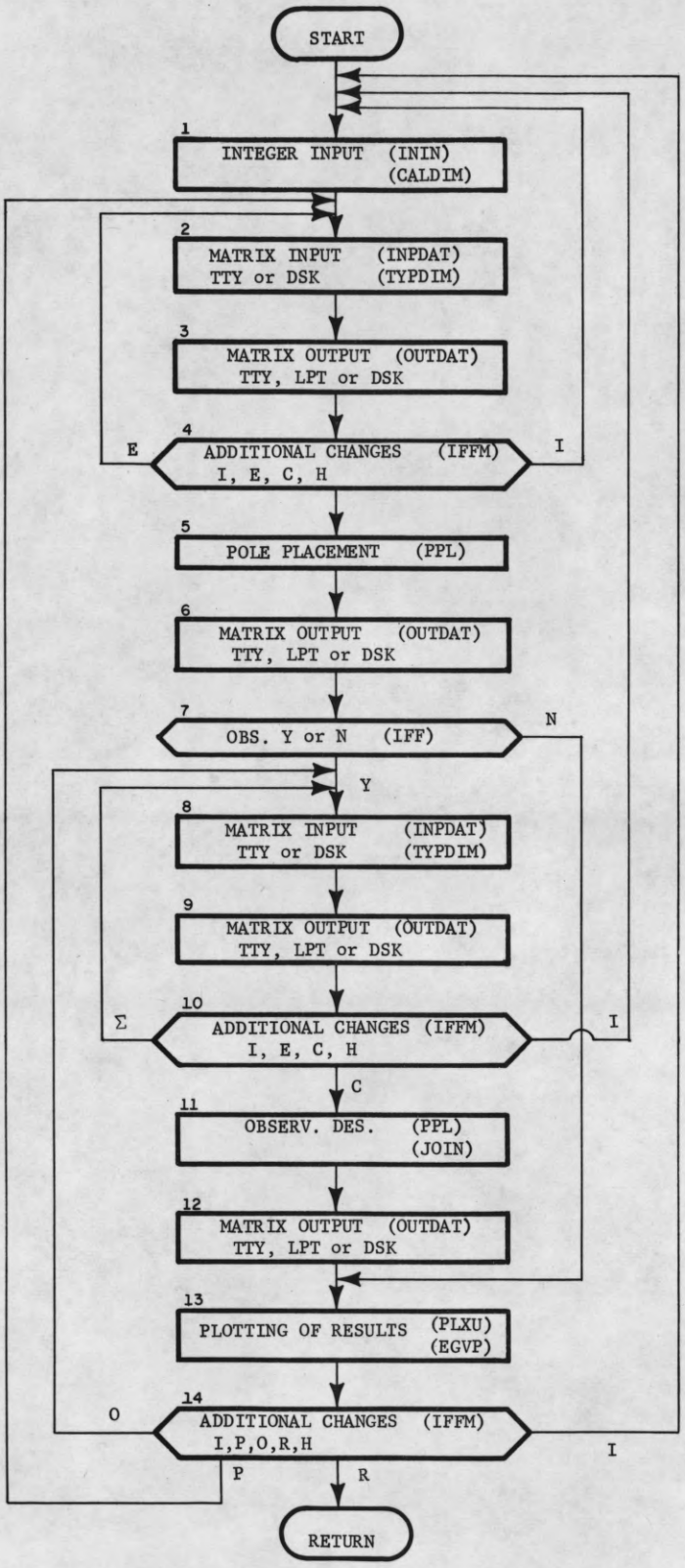

START
1
INTEGER INPUT (ININ)
(CALDIM)
2
MATRIX INPUT (INPDAT)
TTY or DSK (TYPDIM)
3
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
4
E
ADDITIONAL CHANGES (IFFM)
I, E, C, H
I
5
POLE PLACEMENT (PPL)
6
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
7
OBS. Y or N (IFF)
N
Y
8
MATRIX INPUT (INPDAT)
TTY or DSK (TYPDIM)
9
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
10
Σ
ADDITIONAL CHANGES (IFFM)
I, E, C, H
I
C
11
OBSERV. DES. (PPL)
(JOIN)
12
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
13
PLOTTING OF RESULTS (PLXU)
(EGVP)
14
O
ADDITIONAL CHANGES (IFFM)
I,P,O,R,H
I
P
R
RETURN
FP-4467

Figure 4

5. PACKAGE MANAL

BACKGROUND

As stated before, in contrast to design-oriented packages MRIC and MPPL, this package is analysis oriented. Given matrices A, B, C, F, T_m and the scalar Δt, MANAL permits:

a) calculation of the closed-loop matrix

$$A_f = A + BFC \tag{5.1}$$

b) calculation and plotting of eigenvalues of A_f

c) controllability and observability tests.

In order to perform controllability and observability tests a sufficiently small positive number e_c, defined by

$$e_c = 10^{i_c} \ll 1; \qquad i_c < 0$$

should be defined.

d) Calculation of a discrete realization $\{\phi, G, H\}$ corresponding to $\{A_f, B, C\}$, where

$$\begin{aligned} \phi &= \exp\{A_f \Delta t\} \\ G &= A_f^{-1}(\phi - I)B \\ H &= C \end{aligned} \tag{5.2}$$

e) calculation of coefficients of either continuous

$$T(s) = C(Is - A_f)^{-1} B = \{t_{ij}(s)\}, \quad i \in [1,k]; \; j \in [1,m] \tag{5.3}$$

or discrete

$$T(z) = H(Iz - \phi)^{-1} G = \{t_{ij}(z)\} \qquad (5.4)$$

transfer function matrix.

In (5.3) and (5.4) each element $t_{ij}(.)$ is of the form

$$t_{ij}(.) = t^*_{ij}(.)/q(.),$$

$q(.)$ being the characteristic polynomial of either A_f or ϕ.

f) Plotting time-domain responses $x(t)$, $y(t)$ and $u(t)$ of the analyzed system, see Fig. 2.

g) Calculation of a state space realization either in the Jordan form or obtained by applying the given transformation matrix T_m.

All obtained results may be either typed on TTY printed on LPT or stored on DSK for future use within LINSYS. The selection of the particular choice, listed under a),...g), is done in the conversational mode. Undesired options may be easily bypassed by typing an appropriate reply to a computer generated message. As it is the case in the other two packages, all computer generated messages are equipped with the reply 'H', where 'H' stands for Help, permitting assistance in selecting the desired option.

FLOWCHART DESCRIPTION

The first three blocks are basically the same as corresponding blocks in MRIC and MPPL.

The execution of the package MANAL starts by typing on TTY the message

YOU ARE NOW ENTERING INTO THE ANALYSIS PACKAGE
ENTER N,M,K,IC

Block #1. Integers N, M, K and IC corresponding to previously defined

$$n, m, k \text{ and } i_c$$

are entered via TTY. Default values for M, K and IC are

$$M = 1$$

$$K = 1$$

$$IC = -2.$$

Block #2. The message

```
AVAILABLE NAMES ARE ⟨A  ⟩⟨B  ⟩⟨C  ⟩⟨F  ⟩⟨TM ⟩⟨DT ⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Matrices A, B, C, F, TM and the scalar DT are entered either via TTY or DSK. Names and dimensions of entered matrices are typed on TTY by the subroutine TYPDIM. An example of this message is

```
  A       B       C       F       TM      DT
⟨ 5, 5⟩ ⟨ 5, 2⟩ ⟨ 3, 5⟩ ⟨ 2, 3⟩ ⟨ 5, 5⟩ ⟨ 1, 1⟩
```

TM and DT correspond to the transformation matrix T_m and time interval Δt, respectively. If some of these entries are not entered, zero values are assumed.

Block #3. The message

```
AVAILABLE NAMES ARE ⟨A  ⟩⟨B  ⟩⟨C  ⟩⟨F  ⟩⟨TM ⟩⟨DT ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Values of A, B, C, F, TM and DT are either typed on TTY, printed on LPT or stored on DSK.

Block #4. The message

```
TYPE ⟨I⟩,⟨E⟩,⟨C⟩ OR HELP
```

is typed on TTY. Replies 'I', 'E' and 'C' transfer the control to blocks #1, #2 and #5, respectively. In the case of the reply 'H' the following message is typed on TTY:

```
TO CHANGE INTEGERS (MATRIX DIMENSIONS , ETC.);
         ELEMENTS OF MATRICES; OR TO
         CONTINUE
 TYPE ⟨I⟩,⟨E⟩,⟨C⟩ OR HELP
```

Block #5. Closed-loop system matrix A_f and eigenvalues of A_f are calculated. The message

```
DO YOU WANT LOCATIONS OF EIG.VAL. Y OR N
```

is typed on TTY. The subroutine EGVP plots eigenvalue locations on the terminal screen.

Block #6. The message

```
AVAILABLE NAMES ARE ⟨AF ⟩⟨RR ⟩⟨RI ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Matrix AF and arrays RR and RI are either typed on TTY, printed on LPT or stored on DSK. Matrix AF corresponds to the closed-loop matrix A_f, while RR and RI represent real and imaginary parts of the eigenvalues of A_f.

Block #7. The subroutine IFF types on TTY the message

WANT CONTR./OBS. TESTS ?

Reply 'Y' (stands for Yes) transfers the control to the block #8, while the reply 'N' bypasses the block #8 and transfers the control to the block #9.

Block #8. Controllability and observability tests of the triple $\{A_f,B,C\}$ are performed. See the subroutine RANKR. If the triple is both controllable and observable the following messages are typed on TTY

SYSTEM IS CONTROLLABLE
SYSTEM IS OBSERVABLE

Otherwise, the subroutine indicates which mode is either uncontrollable or unobservable, which permits conclusion about the stabilizability and detectability of $\{A_f,B,C\}$. In an example of a stabilizable but not detectable system the following messages are typed on the TTY:

```
MODE   -0.200E+01  +/- J    0.000E+00  NOT CONTROLLABLE
MODE    0.300E+01  +/- J    0.000E+00  NOT OBSERVABLE
```

Block #9. The subroutine IFFM types on TTY the message

TYPE ⟨D⟩,⟨C⟩,⟨N⟩ OR HELP

In the case of the reply 'H' (or 'HELP') the additional information

```
TO OBTAIN DISCRETE TRANSF.FUN.MATRIX'  (STATE TRANSIT.MAT);
          CONTIN.  TRANSF.FUN.MATRIX; OR
          NONE
 TYPE ⟨D⟩,⟨C⟩,⟨N⟩ OR HELP
```

is typed on TTY. As indicated on the flowchart replies 'D', 'C' and 'N' transfer the control to the blocks #10, #11 and #13 respectively.

Block #10. The subroutine DREAL calculates discrete realization (5.2) corresponding to the continuous triple $\{A_f,B,C\}$.

Block #11. The subroutine MTF calculates the coefficients of the characteristic polynomial and transfer function matrix of a given triple. If this block is reached from the block #9 by the reply 'C', then the continuous transfer function matrix (5.3) is calculated. Otherwise, coefficients of the discrete transfer function matrix (5.4) are calculated.

Block #12. The message

AVAILABLE NAMES ARE ⟨CHE⟩⟨TFM⟩⟨F ⟩⟨G ⟩
TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP

is typed on TTY. The subroutine OUTDAT either types on TTY, prints on LPT or stores on DSK the results of the transfer function matrix calculations. F and G correspond to either pair

$$\{A_f,B\} \quad \text{or} \quad \{\phi,G\}$$

depending on whether the block #11 is entered directly by the reply 'C', or after performing the calculations in the block #10. In both cases the entries CHE and TFM contain coefficients of the characteristic polynomial and transfer function matrix, of the corresponding realization, respectively. CHE contains coefficients in ascending order, while TFM is an $(n \times mk)$ array where each column contains n coefficients of a particular element t^*_{ij}

of the transfer function matrix, i.e.

$$TFM = \begin{vmatrix} t^*_{111} & t^*_{121} \cdots & t^*_{1m1} & t^*_{211} & & t^*_{km1} \\ \vdots & \vdots & & \vdots & & \\ t^*_{11n} & t^*_{12n} & t^*_{1mn} & t^*_{21n} & \cdots\cdots & t^*_{kmn} \end{vmatrix}$$

Block #13. The message

WANT X(T),Y(T) & U(T) - Y OR N

is typed on TTY. As in other packages, the subroutine PLXU plots closed-loop responses to initial conditions and polynomial input signal. Used within MANAL, the subroutine PLXU besides state and control vectors x(t) and u(t), plots also the output vector y(t). Selection of the time interval T, initial conditions x^o, dimensions and elements of the matrix E, defined by the block diagram, Fig. 2, is done within the subroutine PLXU.

Block #14. Subroutine IFFM types on TTY the message

TYPE ⟨J⟩,⟨A⟩,⟨N⟩ OR HELP

In the case of the reply 'H' (or 'HELP') the following additional information is typed on TTY

```
TO OBTAIN JORDAN FORM; TO APPLY
          ARBITRARY STATE TRANSFORMATION; OR
          NONE
 TYPE ⟨J⟩,⟨A⟩,⟨N⟩ OR HELP
```

Replies 'J', 'A' and 'N' transfer the control to blocks #15, #16 and #18, respectively.

Block #15. Given the triple $\{A_f,B,C\}$ the subroutine JFORM calculates transformation matrix T_j and a triple $\{A_t,B_t,C_t\}$ given by

$$\begin{aligned} A_t &= T_j^{-1} A_f T_j \\ B_t &= T_j^{-1} B \\ C_t &= C\, T_j \end{aligned} \tag{5.5}$$

where A_t is in the Jordan form.

Block #16. Given the realization $\{A_f,B,C\}$ and the transformation matrix T_m, entered in the block #2, the subroutine TRANS calculates the transformed triple $\{A_t,B_t,C_t\}$ defined by

$$\begin{aligned} A_t &= T_m^{-1} A_f T_m \\ B_t &= T_m^{-1} B \\ C_t &= C\, T_m. \end{aligned} \tag{5.6}$$

Block #17. The message

```
AVAILABLE NAMES ARE ⟨AT ⟩⟨BT ⟩⟨CT ⟩⟨FT ⟩⟨TM ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩,⟨N⟩ OR HELP
```

is typed on TTY. Some matrices among

AT, BT, CT, FT and TM

are either typed on TTY, printed on LPT or stored on DSK. Matrices AT, BT and CT correspond either to (5.5) or (5.6). Matrix FT = F, while transformation matrix TM is either equal to T_j or T_m, depending on whether

in the block #14, the option 'J' or 'A' is used, respectively.

Block #18. The message

TYPE ⟨I⟩,⟨E⟩,⟨P⟩,⟨R⟩ OR HELP

is typed on TTY. Similarly to last blocks in MRIC and MPPL, this block transfers the control to either blocks #1, #2 or #13 within MANAL (replies 'I', 'E' or 'P') or returns to the block #1 of LINSYS (reply 'R'). In the case of the reply 'H' the following message

```
TO CHANGE INTEGERS (MATRIX DIMENSIONS, ETC.);
          ELEMENTS OF MATRICES; TO OBTAIN NEW
          PLOTS; OR TO
          RETURN TO LINSYS
 TYPE ⟨I⟩,⟨E⟩,⟨P⟩,⟨R⟩ OR HELP
```

is typed on TTY.

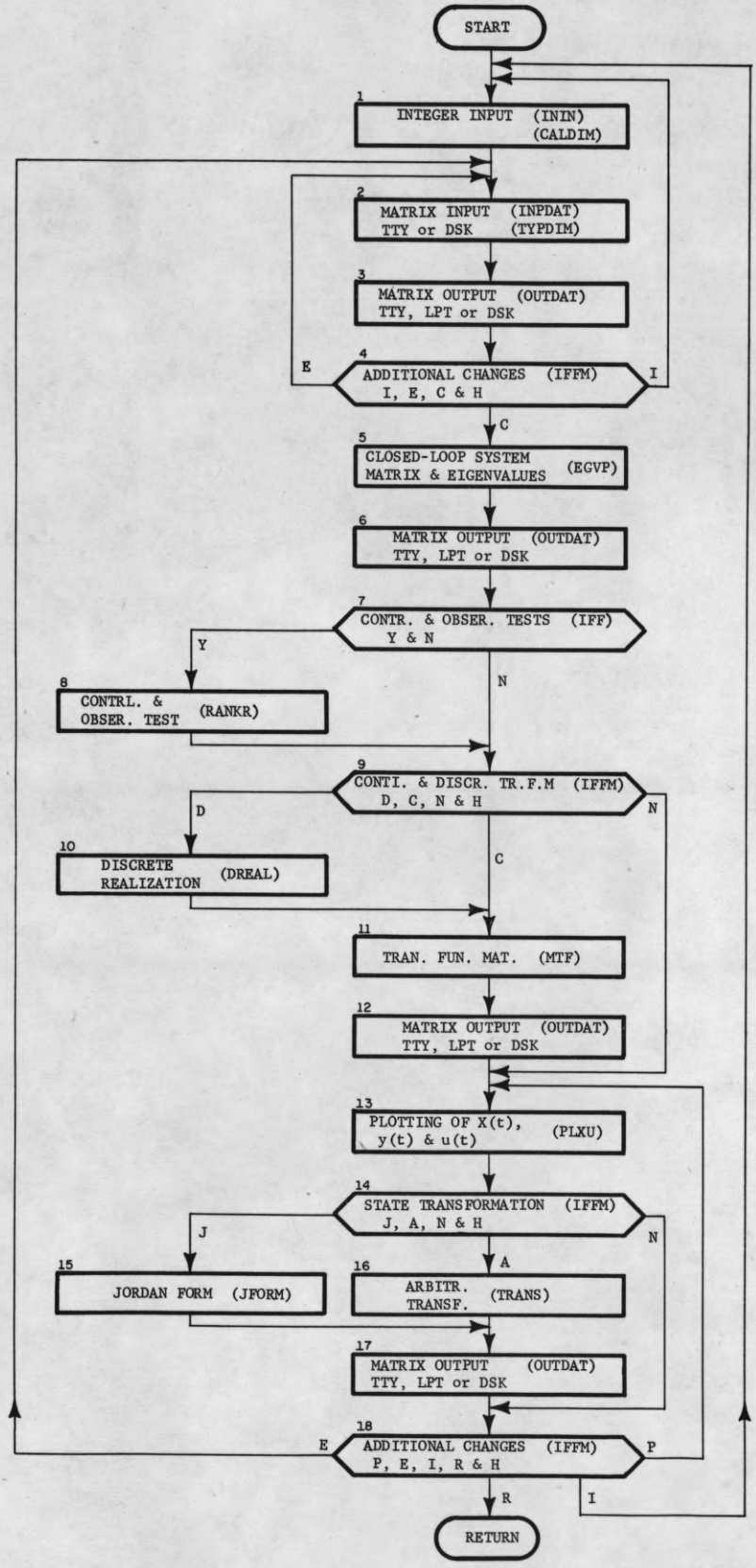

START
1
INTEGER INPUT (ININ)
(CALDIM)
2
MATRIX INPUT (INPDAT)
TTY or DSK (TYPDIM)
3
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
4
E
ADDITIONAL CHANGES (IFFM)
I, E, C & H
I
C
5
CLOSED-LOOP SYSTEM
MATRIX & EIGENVALUES (EGVP)
6
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
7
CONTR. & OBSER. TESTS (IFF)
Y & N
Y
N
8
CONTRL. &
OBSER. TEST (RANKR)
9
CONTI. & DISCR. TR.F.M (IFFM)
D, C, N & H
D
N
C
10
DISCRETE
REALIZATION (DREAL)
11
TRAN. FUN. MAT. (MTF)
12
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
13
PLOTTING OF X(t),
y(t) & u(t) (PLXU)
14
STATE TRANSFORMATION (IFFM)
J, A, N & H
J
N
A
15
JORDAN FORM (JFORM)
16
ARBITR.
TRANSF. (TRANS)
17
MATRIX OUTPUT (OUTDAT)
TTY, LPT or DSK
18
E
ADDITIONAL CHANGES (IFFM)
P, E, I, R & H
P
R
I
RETURN
FP-4468

Figure 5

6. PACKAGE EDITD

The package EDITD is used for editing disk data file accessible by the software LINSYS. Execution of the EDITD package requires the assignment of disks #2, #3 and #4. Automatic disk assignment may be done by including into the user's disk area the following file

```
SWITCH.INI
LOGIN/ASSIGN:DSK:2/ASSIGN:DSK:3/ASSIGN:DSK:4
```

The flowchart of the package EDITD is given in Fig. 6. EDITD permits:

(a) either typing on TTY or printing on LPT names and numbers of existing records - subroutine PNAMES - block #2,

(b) either typing on TTY or printing on LPT record names, dimensions and elements of all stored matrices - subroutine PRIDAT - block #3, and

(c) changing elements in already stored matrices, introduction of new records and deletion of existing records - block #4.

The package begins execution by typing on TTY the following message

```
TYPE ⟨N⟩,⟨M⟩,⟨D⟩,⟨S⟩ OR HELP
```

In the case of the reply 'H', or HELP, the following additional information is typed on TTY:

```
TO TYPE/PRINT NAMES;
              MATRICES;
              DELETE/INSERT/CHANGE RECORDS; TO
              STOP; OR FOR
              HELP
 TYPE ⟨N⟩,⟨M⟩,⟨D⟩,⟨S⟩ OR HELP
```

The user selects a particular option by responding on the TTY with 'N', 'M', 'D' or 'S'.

Blocks #2 and #3 start their execution by typing on TTY the message

TYPE ⟨L⟩ OR ⟨T⟩

In the case of the reply 'T' the results are displayed on the user's terminal, while in the case of the reply 'L' LPT is used.

Selection of a particular option in the block #4 is done in the conversational mode within the block. If the user has deleted some of the existing records, then before exiting from the EDITD on TTY the following message is typed:

```
TYPE: DEL NAMES.DAT,MATRIC.DAT
      REN NAMES.DAT=TNAM.DAT                    (6.1)
      REN MATRIC.DAT=TMAT.DAT
```

In other words, in this case, after exiting from the package the following monitor commands should be typed

```
.DEL NAMES.DAT,MATRIC.DAT
.REN NAMES.DAT=TNAM.DAT                         (6.2)
.REN MATRIC.DAT=TMAT.DAT
```

by which the edited data from temporary files (TNAM.DAT and TMAT.DAT) will be transferred to files NAMES.DAT and MATRIC.DAT accessible by the subroutines INPDAT and OUTDAT of the LINSYS.

In the case of using block #4 only for entering new records or for changing matrix elements in already existing records, i.e. when package does not type message (6.1), the user is warned not to type commands (6.2) because in this case the whole data file will be destroyed. An example of EDITD execution is given in Sec. 16.

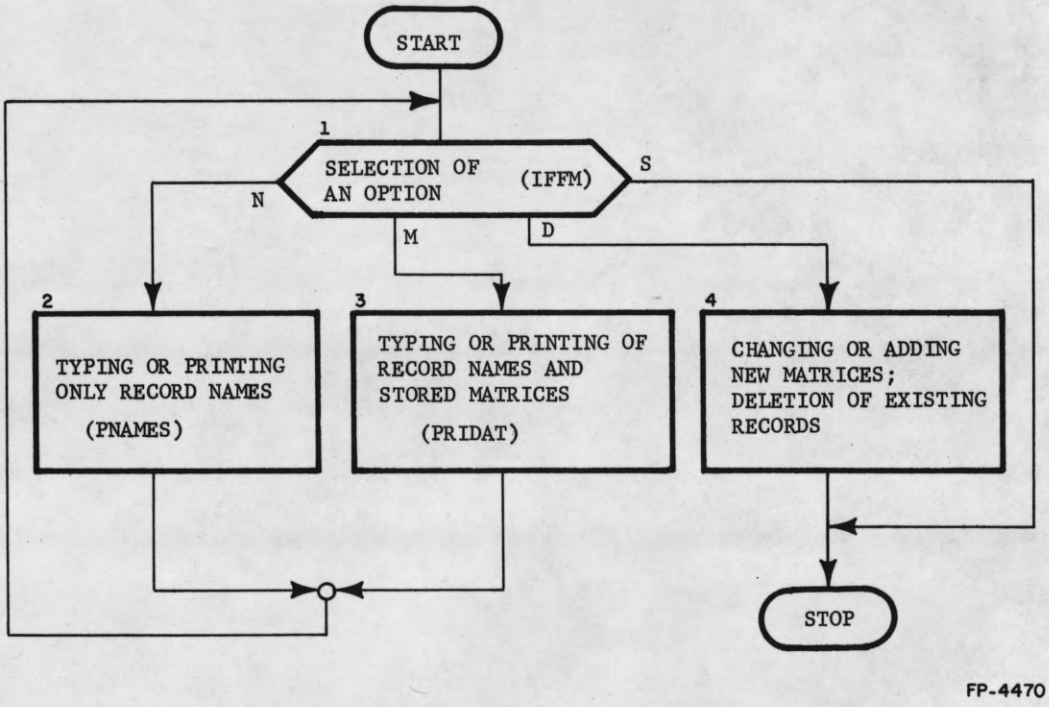

START
1
SELECTION OF AN OPTION (IFFM)
N
M
D
S
2
TYPING OR PRINTING ONLY RECORD NAMES (PNAMES)
3
TYPING OR PRINTING OF RECORD NAMES AND STORED MATRICES (PRIDAT)
4
CHANGING OR ADDING NEW MATRICES; DELETION OF EXISTING RECORDS
STOP
FP-4470

Figure 6

7. PLOTTING SUBROUTINES

7.1. Subroutine PLXU

The subroutine PLXU is used for calculating and plotting on the terminal screen the system responses to initial conditions and polynomial input signal. See block diagram in Fig. 2. Subroutine PLXU is called within all packages MRIC, MPPL and MANAL.

Calling sequence

```
CALL PLXU(AF,B,F,C,N,M,K,IND)
```

Definition of arguments

AF = n x n matrix specifying closed loop system matrix A_f

N,M,K = integers specifying system order n, number of inputs m, and number of outputs k, respectively.

IND = identifier, 2 character ASCII variable = {1HRC, 1HPP, 1HPO, 1HAN}

Identifier IND is used to accommodate different requirements imposed by pacakges MRIC, MPPL and MANAL in plotting input and output signals u(t) and y(t), respectively. See subroutine CALYU used in calculating u(t) and y(t). The definition of other arguments B, F and C highly depends on the value of the identifier IND. They will be explained in conjunction with the subroutine CALYU.

FLOWCHART DESCRIPTION

Block #1. Entries ND, E, XO and TT, corresponding to integer n_d, $(n \times n_d)$ disturbance matrix E, n dimensional initial condition vector x^o, and time interval T, respectively are entered by the subroutine INPDAT. Names and dimensions of these entries are typed on TTY.

Block #2. Numeric values for ND, E, XO and TT are either typed on TTY, printed on LPT or stored on DSK by the subroutine OUTDAT.

Block #3. Subroutine RESP calculates the state vector defined by:

$$\dot{x}(t) = A_f x(t) + w(t); \qquad t \in [0,T] \; ; \quad x(0) = x^o$$

$$w(t) = \sum_{i=1}^{n_d} e^i t^{i-1} = E\, z(t), \tag{7.1}$$

$$z(t) = \left|1 \quad t \quad t^2 \quad .... \quad t^{n_d-1}\right|^T.$$

For details see subroutine RESP.

Block #4. Entries ND, E, XO, TT and RES are either typed on TTY, printed on RTL or stored on DSK. RES is a $(n \times k)$ matrix containing $x_i(t_j)$, $i \in [1,n]$, $j \in [1,k]$, $t_j = (j-1)\,\Delta t$, $\Delta t = T/(k-1)$, $k = [300/n]$.

Block #5. Subroutine SEL performs selection of indices of the state vector x(t) to be plotted. By the subroutine SEL the following message is typed on TTY:

TYPE INDICES YOU WANT TO PLOT

The desired indices are typed format-free separated by commas. Typing the

'RETURN' key or zero bypasses the plotting. Typing any integer greater than n causes plotting of all n elements of the vector x(t).

Block #6. Subroutine PLOTN plots on the terminal screen the selected elements of x(t). In order to achieve maximum use of the available space of the screen, the Y-axis of each plot is scaled differently. After completing the plot the subroutine PLOTN types on TTY the message

DO YOU WANT NEW PLOT WITH THE COMMON SCALE-Y OR N

In the case of the reply 'Y' the subroutine PLOTN types on TTY:

YMIN,YMAX

where YMIN and YMAX define the common scale to be used in the new plot. The user types values for YMIN and YMAX format-free, separated by the comma. The reply 'N' transfers the control to the next block.

Block #7. Depending on the identifier IND, i.e. depending on the particular package calling the subroutine PLXU, the subroutine CALYU calculates either u(t) or both y(t) and u(t). For details refer to the subroutine CALYU.

Block #8. Similarly as the block in the block #5, the subroutine SEL selects the indices of elements of u(t) and y(t) to be plotted on the terminal screen.

Block #9. Similarly as in the block #6, the subroutine plots the selected elements of y(t) and u(t) with either different or both different and common scale.

Block #10. The subroutine IFFM plots on TTY the message

TYPE ⟨E⟩,⟨I⟩,⟨N⟩ OR HELP

In the case of the reply 'H' (or HELP) the following additional information is typed on TTY

```
TO CHANGE ELEMENTS IN ND,E,XO AND TT;
          INDICES; OR
          NONE
 TYPE ⟨E⟩,⟨I⟩,⟨N⟩ OR HELP
```

The replies 'E' and 'I' transfer the control to blocks #1 and #5, of PLXU while the reply 'N' will transfer the control to the calling program, i.e. to one of the packages MRIC, MPPL or MANAL.

An example of user-computer conversation within the subroutine PLXU is given in Sec. 16.

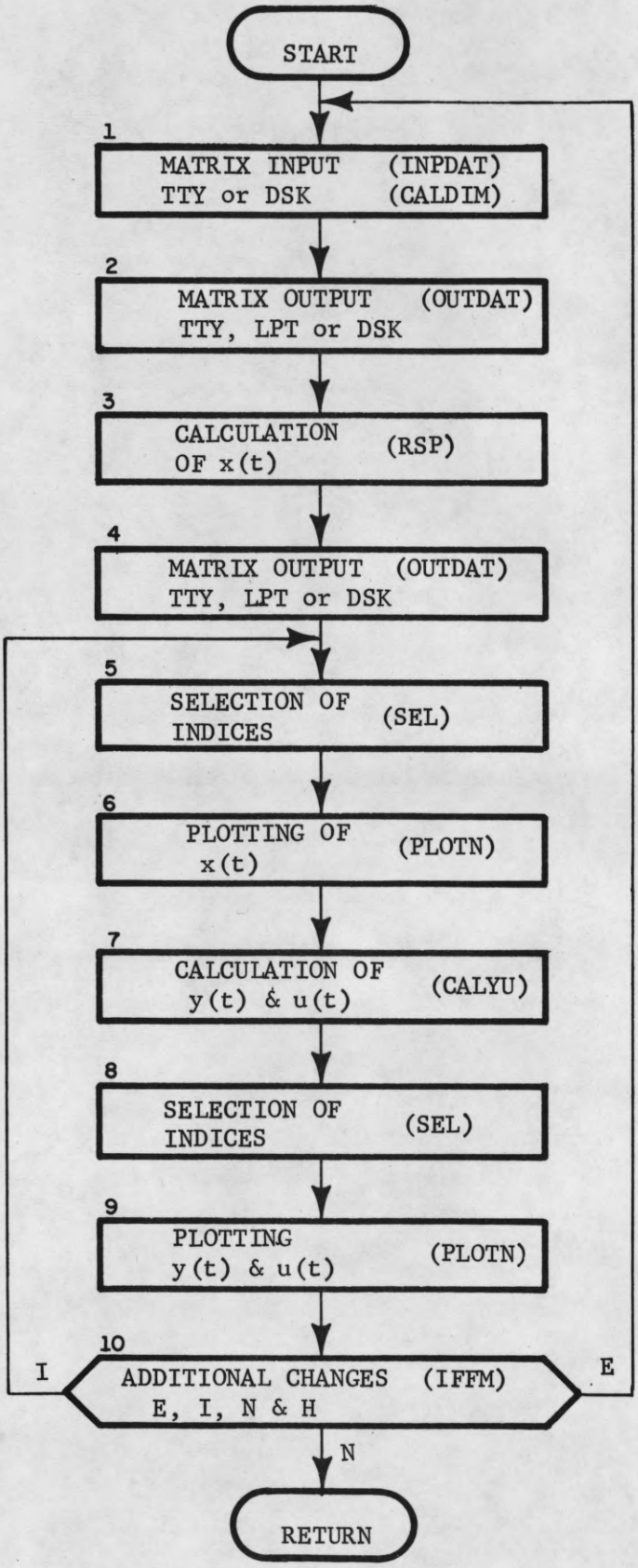


Figure 7

FP-4471

7.2. Subroutine EGVP

The subroutine EGVP is used for calculation of eigenvalues of a given (n X n) matrix and for plotting a part of the s-plane containing either all or only dominant eigenvalues.

Calling sequence

```
CALL EGVP(A,N,RR,RI,IND)
```

Definition of arguments

A = given (n X n) matrix

N = integer specifying the order n of the matrix A

RR, RI = N dimensional arrays containing real and imaginary parts of eigenvalues, respectively

IND = identifier; for IND ≠ 0 eigenvalue plotting is bypassed.

FLOWCHART DESCRIPTION

Block #1. Arrays RR and RI are calculated.

Block #2. Subroutine MRLOC plots a part of the s-plane containing all eigenvalue locations.

Block #3. The message

DO YOU WANT NEW PLOT

is typed on TTY. Reply 'Y' transfers the control to the block #4, while the reply 'N' causes return to the calling program.

Block #4. The message

WHICH EIGENVAL EIG.VAL. YOU WANT TO ELIMINATE
TYPE LIM.VALUE

is typed on TTY followed by values of all elements of the array RR. After user types the limiting value RR.LIM, the subroutine MRLOC plots locations of only those eigenvalues whose real parts are greater than the entered RR.LIM.

An example of user-computer conversation within the subroutine EGVP is given in Sec. 16.

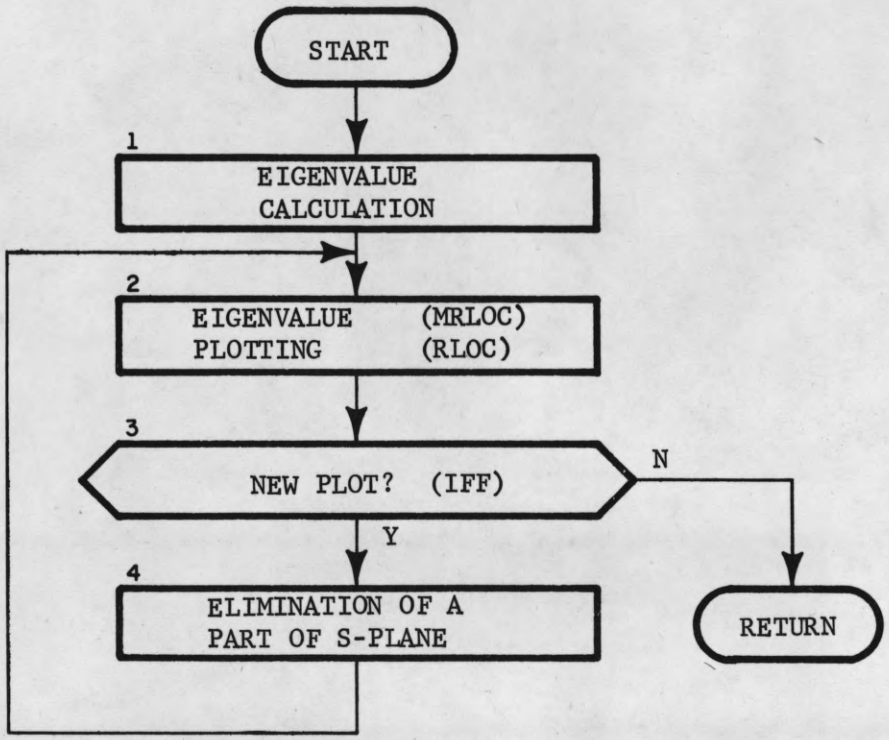

START
1
EIGENVALUE
CALCULATION
2
EIGENVALUE (MRLOC)
PLOTTING (RLOC)
3
NEW PLOT? (IFF)
N
Y
4
ELIMINATION OF A
PART OF S-PLANE
RETURN
FP-4472

Figure 8

7.3. Subroutine BLDG

The subroutine BLDG is used for plotting the system block diagram, given on Fig. 2.

Calling sequence

```
CALL BLDG
```

The calling sequence does not have any arguments. Within the subroutine BLDG the following subroutines

SQ, LN, AR and CR

are called. They are used for plotting rectangles, straight lines, arrows and circles, respectively.

The write-ups of these subroutines are attached to the write-up of the subrouting BLDG. The letters and special characters as A,B,...,(,/,), e etc. are plotted by the subroutine NOTATE which is a part of the graphic software AG-II.

Calling sequences for these subroutines are

```
CALL SQ(X1,Y1,DX,DY)
```

Definition of arguments

X1 and Y1 = integers specifying (x,y) coordinate of the lower left corner of the rectangle

DX and DY = integers specifying lengths in x and y directions, respectively.

CALL LN(X1,Y1,X2,Y2)

Definition of arguments

X1 and Y1 = integers specifying starting point of the straight line

X2 and Y2 = integers specifying final point of the straight line

CALL AR(J1,J2,N,IND)

Definition of arguments

J1 and J2 = integers specifying (x,y) coordinate

N = integer specifying displacement

IND = identifier = $\left\{\begin{matrix}'R'\\'L'\\'U'\\'D'\end{matrix}\right\}$

Depending on the value of identifier IND the subroutine plots the arrow either to the right, to the left, upwards or downward, respectively.

The role of the displacement N is explained by the diagram in Fig. 9.

CALL CR(I1,I2,N)

I1 and I2 = integers specifying (x,y) coordinate of the center of the circle,

N = integer specifying the radius of the circle.

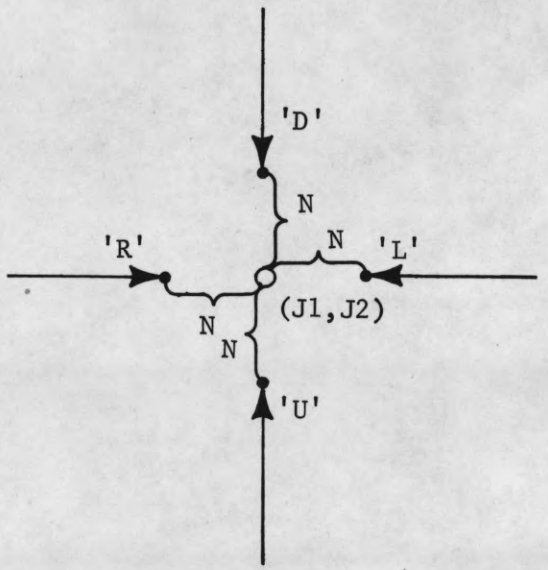


Figure 9

8. DATA HANDLING SUBROUTINES

8.1. Integer Input

Subroutine ININ

Subroutine ININ is used for entering from TTY up to six integers, specifying dimensions of matrices, or some other options in the calculation.

Calling sequence

```
CALL ININ(N,NT,IT,I1,I2,I3,I4,I5,I6)
```

Definition of arguments

N = number of integers to be read, $1 \leq N \leq 6$.

IT = arbitrary text (string of NT alpha-numeric ASCII characters) to be typed on TTY before reading the integers I_i, $i \in [1,N]$.

NT = number of ASCII characters in the text IT, $1 \leq NT \leq 50$.

I1,...I6 = integers to be read from TTY.

The text IT is used to inform the user which integers should be entered by the subroutine. In the case of reading less than six integers, the remaining integers should be dummy integer variables. Example:

```
CALL ININ(3,11,'ENTER N,M,K',N,M,K,L,L,L)
```

In this case, execution of the subroutine will start by typing on the TTY the message

```
ENTER N,M,K                                                    (8.1)
```

Typing the integer values is format-free. As delimiter the comma ',' is used. The reading of these values is done by the subroutine IOMT, see

Sec. 8.2.4. All LINSYS packages start with the subroutine ININ. In order to cancel previous typing which may occur during loading or complication, after typing numerical values, but before assigning the typed values to integers, I1,... I6, the subroutine ININ types on TTY the message

ARE YOU SATISFIED WITH INPUT DATA - Y OR N

In the case of user's reply 'N' the subroutine will again type the message (8.1), while the reply 'Y' will cause the exit from the subroutine. The processing of user's replies 'Y' and 'N' is done by the subroutine IFF, see Sec. 9.

8.2. Matrix Input

Subroutine INPDAT

The subroutine INPDAT is used for reading up to six matrices from either TTY or DSK into the core memory.

Calling sequence

```
CALL INPDAT(N,M,IT,A1,A2,A3,A4,A5,A6,IR)
```

Description of symbols

N = integer specifying number of matrices to be read; $1 \leq N \leq 6$.

A1,... A6 = matrices to be read, stored columnwise.

M = 2N dimensional array defining dimensions of matrices A_i. Elements M(2*I-1) and M(2*I) specify number of rows and columns of the matrix A_i, $i \in [1,6]$, respectively.

IT = N dimensional array containing names for matrices A_i. Each matrix name IT(I) is up to 3-character ASCII variable. The array IT is usually defined by the FORTRAN DATA statement. See write-ups of any package.

IR = Identifier used by the subroutine INPTTY. For IR = 1, options permitting changing of particular element, row, column or main diagonal of an already defined matrix are not available, see subroutine INPTTY. After each call of the subroutine INPDAT the values of the identifier IR is increased by one, see write-up of INPDAT.

The choice of whether matrices will be read from TTY or DSK is done in a conversational mode. A flowchart of the subroutine INPDAT is given in

Fig. 10. The execution of the subroutine starts by typing on TTY all N available matrix names - elements of the array IT - and then by typing the message

TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP (8.2)

An example of typing the available names is

```
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩⟨K  ⟩⟨AOF⟩
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

In the case of the reply 'H', the following message is typed

```
FOR CHANGING OR READING MATRICES FROM TTY, DSK OR NONE
 TYPE ⟨T⟩,⟨D⟩,⟨N⟩ OR HELP
```

The handling of user's replies is done by the subroutine LFFM, see Sec. 9. According to the user's choice, the subroutine INPDAT calls either subroutine INPTTY (reply 'T') or subroutine INPDSK (reply 'D'), by which actual reading is done.

After executing either of INPTTY or INPDSK the subroutine INPDAT again types the message (8.2), allowing the user to enter additional matrices or to change some elements within already defined matrices. Only the reply 'N' will cause the exit from the subroutine.

8.2.1. Subroutines INPTTY and INPDSK

The subroutine INPTTY and INPDSK are used for reading up to six matrices from TTY and DSK respectively.

Calling sequences

```
CALL INPTTY(N,M,IT,A1,A2,A3,A4,A5,A6,IR)
CALL INPDSK(N,M,IT,A1,A2,A3,A4,A5,A6)
```

The definitions of arguments in calling sequences are the same as in the subroutine INPDAT. Either of the subroutines starts its execution by typing on TTY the message

$$\text{TYPE MATRIX NAMES YOU WANT OR TYPE EITHER } \langle N \rangle \text{ OR } \langle ALL \rangle \quad \text{TO} \begin{cases} \text{CHANGE VIA TTY} \\ \text{READ FROM DSK} \end{cases} \tag{8.3}$$

As a reply to the message (8.3) the user is expected to type some or all matrix names (elements of the array IT) previously typed on TTY by the subroutine INPDAT. If he wants to enter all available matrices he may also type the reply 'ALL'. Typing the reply 'ALL' is equivalent to typing all N matrix names. User types desired matrix names format-free. As delimiter the comma ',' is used. Format-free reading of matrix names is performed by subroutine RCHAR, called by either subroutine INPTTY and INPDSK. Typing the reply 'N' causes exit from the subroutine, i.e. return to the calling subroutine INPDAT. If among the matrix names the user has typed a nonexisting name, say 'XAX', either of the subroutines will type the message

MATRIX WITH THE NAME ⟨XAX⟩ DOES NOT EXIST - TRY AGAIN

followed again by the message (8.3), waiting for the correct names to be typed. After selecting the desired matrix names, either subroutine INPTTY and INPDSK calls several times its own subroutine by which reading or changing of elements of each particular matrix is done.

Subroutine INPTTY calls CHANGE and IOMT
INPDSK calls DSKRD and ADISK

The number of calls to these subroutines corresponds to the number of correct matrix names typed in reply to the message (8.3).

After reading the desired number of matrices by either subroutine, control is returned to the calling subroutine INPDAT, which will again type the message (8.2). The user reply 'N' to the message (8.3) causes direct return to the calling subroutine INPDAT.

8.2.2. Changing of Elements of Already Defined Matrices

In addition to these features, subroutine INPTTY also permits changing a particular element, row, column or the main diagonal in already defined matrices. After selecting the desired matrix names, the subroutine INPTTY types the message

IN 'XYZ' YOU WANT TO CHANGE E,R,C,D,M OR N (8.4)

where 'XYZ' is one of the matrix names the user has typed as a reply to the message (8.3). If the user wants to change either particular element, row or column, he has to type either 'E', 'R' or 'C' and in this case the subroutine responds with the message

TYPE INDICES OF EL. ROW OR COL. (8.5)

In reply to the message (8.5) the user has to type indices of the element, row or column he wants to change. Typing of indices is format-free, using comma ',' as the delimiter. If the user wants either to change the main diagonal or to enter an entire new matrix, he has to type either 'D' or 'M'.

The reply 'N' to the message (8.4) leaves the matrix with the name XYZ unchanged, and the subroutine proceeds to the next matrix by typing again the message (8.4) using the name of the next matrix to be changed or entered.

In the case of replying to the message (8.4) by 'H', the following additional information is typed on TTY:

```
E, R, C, M & N STAND FOR
ELEMENT, COLUMN, DIAGONAL (MAIN), MATRIX (WHOLE) & NONE
```

If accidentally, the user has typed a nonexisting option, say 'X', or in reply to the message (8.5) he has typed index/indices of element, row or column which is/are out of bounds, i.e. greater than the number of rows, or columns in the particular matrix, the subroutine INPTTY will respond by either of the messages

```
OPTION ⟨X⟩ DOES NOT EXIST - TRY AGAIN
```

OR

```
INDICES OUT OF BOUNDS
```

and will be ready again to accept a correct option or correct indicies values.

After entering the desired option and, if necessary, indices values, the subroutine INPTTY retypes the matrix name indicating to the user it is ready to accept numerical data for this particular matrix. As it has been mentioned in connection with the subroutine INPDAT, in the case of identifier IR = 1, only entering of all matrix elements is possible. If some of the arguments A1,... A6 represent a scalar, then after the user types the

name, subroutine INPTTY immediately retypes that name expecting its numerical value.

Typing of numerical values is format-free. These data are read by the subroutine IOMT. These data are then stored by the subroutine CHANGE into particular locations of the matrix.

8.2.3. Subroutine CHANGE

Subroutine CHANGE is used for substituting either a particular element, row, column, main diagonal or a whole matrix by other numerical values.

Calling sequence

```
CALL CHANGE(A,AA,IA,I1,I2,N,M)
```

Definition of arguments

A = N X M matrix to be changed,

IA = identifier = 1HE, 1HR, 1HC, 1HD, 1HM,

I1,I2 = indices of element, row or column of the matrix A where the new numerical data are to be stored,

N,M = integers, number of rows and columns of the matrix A,

AA = array of numerical values which according to the option (value of the identifier IA) have to be stored into particular locations of the matrix A.

Subroutine DSKRD

Subroutine DSKRD is used for reading from DSK into core memory a previously stored matrix. Subroutine reads both matrix dimensions and elements.

Calling sequence

CALL DSKRD(A,NV,ITA)

Description of arguments

A = N X M matrix to be read from DSK,

NV = 2-dimensional integer array containing matrix dimensions,

NV(1) = N = number of rows, NV(2) = M = number of columns,

ITA = ASCII variable containing matrix names. ITA is usually an element of the array IT appearing in INPDAT and INPDSK subroutines.

Since all matrices are stored on DSK in separate records holding a particular record name, in order to read a desired matrix from DSK it is necessary to specify the name of the record where the desired matrix is stored. To this end execution of the subroutine DSKRD starts by typing on TTY the message

ENTER THE RECORD NAME FOR MATRIX ⟨A1⟩ (8.6)

where it is assumed that 'A1' represents the matrix name, i.e. ITA = 2HA1.

In order to obtain the desired matrix, the user should remember under which name the desired matrix has been stored on DSK. As a reply to the message (8.6), the user has to type the desired record name. Assume that the user has typed 'A101'. If a record exists on DSK with the name 'A101', the subroutine reads from that record both the dimensions N,M and matrix elements, and transfers them into the arrays NV and A respectively. If a record with the name 'A101' does not exist the subroutine types on TTY the message:

RECORD WITH NAME ⟨A101⟩ DOES NOT EXIST

followed immediately by the message (8.6), expecting that the user will type one of the existing record names. For the user's information after successful reading, the subroutine will type on TTY the message:

MATRIX ⟨A101⟩ FOUND IN REC # n_1

where n_1 represents record number associated with the name 'A101'. Using for a record name the following string of 5 characters

SKIP.

causes direct return to the calling subroutine INPDSK.

8.2.4. Subroutine IOMT

The subroutine IOMT is used for both format-free reading or typing a N X M matrix on TTY. In the case of typing the subroutine uses either format 10F6.2 or 5E12.4.

Calling sequence

```
CALL IOMT(A,N,M,IND,NT,IT)
```

Definition of arguments

A = N X M matrix, stored columnwise,

N = integer, number of rows in A,

M = integer, number of columns in A,

IND = identifier = -2, -1, 0, 1, 2, and 3, -3

IT = arbitrary text - string of NT-ASCII alpha-numeric characters to be typed on TTY before reading or writing matrix elements.

NT = integer specifying number of ASCII characters in the text IT,

$1 \leq NT \leq 50$.

Examples:

```
CALL IOMT(XO,1,N,-1,17,'INITIAL CONDITIONS')
CALL IOMT(B,M(J-1),M(J),2,3,IT(K))
```

The role of the identifier IND is explained by the following table

IND	READING	TYPING WITH FORMAT 10F6.2	TYPING WITH FORMAT 5E12.4
-2	yes	no	yes
-1	yes	yes	no
0	yes	no	no
1	no	yes	no
2	no	no	yes
3	no	the use of format depends on numerical values of matrix elements	
-3	yes	" " " " "	

In the cases of IND = 3 and IND = -3 the format 10F6.2 will be used only if all matrix elements are within the limits

$$-99.99 \leq A_{ij} \leq 999.99$$

otherwise the format 5E12.4 will be used.

Format-free reading feature of the subroutine IOMT requires the use of the comma ',' as delimiter between matrix elements. If a matrix element has zero value it is sufficient to type only the comma. Example: In order to enter the 4 X 4 matrix

$$A = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.1 & 0 & 0 & 1 \\ -24 & -50 & -35 & -10 \end{vmatrix}$$

it is sufficient to type

```
,1
,,1
.1,,,1
-24,-50,-35,-10
```

although it is possible to enter the same matrix in the following way

```
0,1.,0,0,
0,0,1,0
0.1,0,0,1
-24,-50.0,-35,-10,
```

If in typing matrix elements the user makes a typing error, e.g.

instead of -100	he types	--10
instead of 3.5,1.5	he types	3.5.1.5
instead of 1,2	he types	A,2

the subroutine IOMT with response with messages

```
I BEG YOUR PARDON     --10
I BEG YOUR PARDON     3.5.1.5                    (8.7)
I BEG YOUR PARDON     A,
```

respectively, will allow the user to type again the same row. If the user types in a single row more elements than it is specified by the integer M (number of columns in matrix A) the subroutine will respond by the message

TOO MANY ELEMENTS IN THE ROW - TRY AGAIN (8.8)

For example, this message will occur if in response to the statement

```
CALL IOMT(XO,1,3,-1,17,'INITIAL CONDITION')
```

the user types

```
1,,,2
```

instead of

```
1,,2
```

The checking for typing errors and typing the messages (8.7) and (8.8) is done by subroutines IM and INTT called within the subroutine IOMT. See listings of these subroutines.

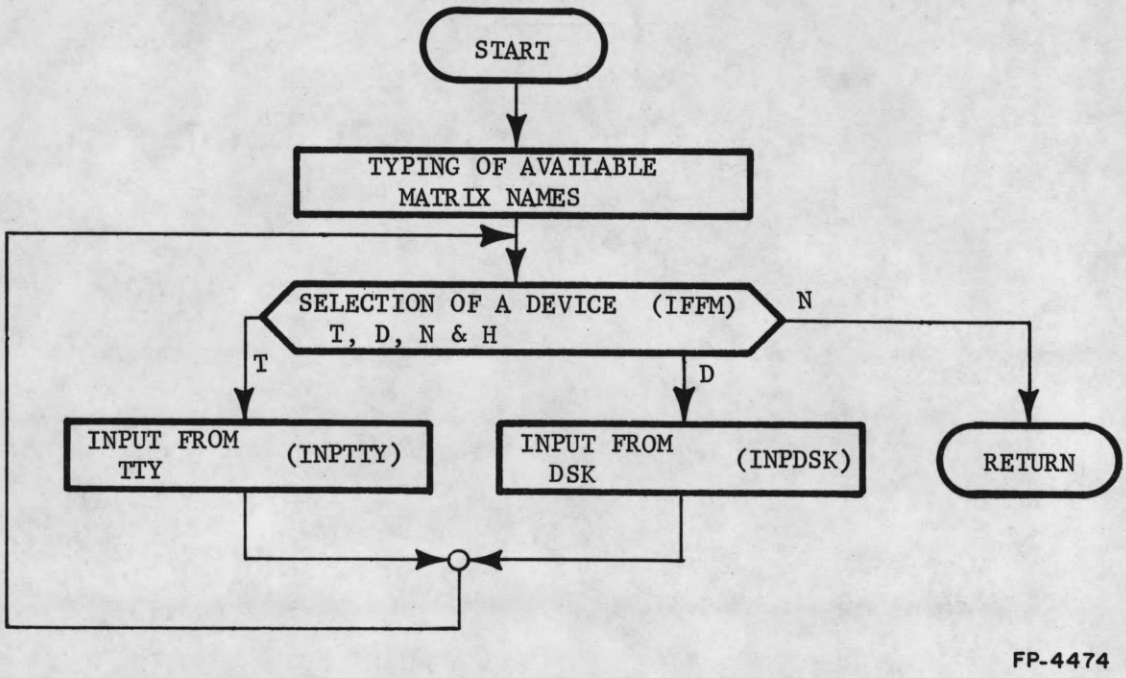

START
TYPING OF AVAILABLE MATRIX NAMES
SELECTION OF A DEVICE (IFFM)
T, D, N & H
N
T
D
INPUT FROM TTY (INPTTY)
INPUT FROM DSK (INPDSK)
RETURN
FP-4474

Figure 10

8.3. Matrix Output

Subroutine OUTDAT

The subroutine OUTDAT is used for writing up to six matrices from core memory into either TTY, LPT or DSK.

Calling sequence

```
CALL OUTDAT(N,M,IT,A1,A2,A3,A3,A4,A5,A6)
```

N = integer specifying number of matrices to be written, $1 \leq N \leq 6$.

A1,... A6 = matrices to be written, stored columnwise.

M = 2N dimensional array, defining dimensions of matrices A_i. Elements M(2*i-1) and M(2*i) specify number of rows and columns of the matrix A_i, $i \in [1,N]$,

IT = N dimensional array containing 3-character names for matrices A_i.

The flow chart of the subroutine OUTDAT is given in Fig. 11. The execution of the subroutine starts by typing on TTY all N available matrix names - elements of the array IT - and then by typing the message

```
TYPE ⟨T⟩,⟨L⟩,⟨D⟩ OR HELP                                    (8.9)
```

An example of typing the available names is

```
AVAILABLE NAMES ARE ⟨C  ⟩⟨RO ⟩⟨IO ⟩
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩ OR HELP
```

In the case of the reply 'H' the following message is typed

```
FOR WRITING MATRICES ON TTY, LPT, DSK, OR NONE
 TYPE ⟨T⟩,⟨L⟩,⟨D⟩ OR HELP
```

The choice whether matrices will be written on TTY, LPT or DSK is done in a conversational mode within the subroutine OUTDAT. The handling of the user's replies is done by the subroutine IFFM, see Sec. 9.

According to the user's choice, the subroutine OUTDAT calls either OUTTTY (reply 'T'), OUTLPT (reply 'L') or OUTDSK (reply 'D'), by which actual writing is done. After executing either of OUTTTY, OUTLPT or OUTDSK the subroutine OUTDAT again types the message (8.9), allowing the user to write or store additional matrices. Only the reply 'N' causes the exit from the subroutine.

8.3.1 Subroutines OUTTTY, OUTLPT and OUTDSK

These subroutines are used for writing/storing up to six matrices into TTY, LPT and DSK, respectively.

Calling sequences

```
CALL OUTTTY(N,M,IT,A1,A2,A3,A4,A5,A6)
CALL OUTLPT(N,M,IT,A1,A2,A3,A4,A5,A6)
CALL OUTDSK(N,M,IT,A1,A2,A3,A4,A5,A6)
```

The definitions of arguments in calling sequences are the same as in the subroutine OUTDAT (see Sec. 8.3). Each of the subroutine starts its execution by typing on TTY the message

$$\text{TYPE NAMES OF MATRICES YOU WAN TO } \left\{ \begin{array}{l} \text{TYPE ON TTY} \\ \text{PRINT ON LPT} \\ \text{STORE ON DSK} \end{array} \right\} \qquad (8.10)$$

OR TYPE EITHER ⟨N⟩ OR ⟨ALL⟩

As a reply to the message (8.10), the user is expected to type some or all matrix names (elements of the array IT) previously typed on TTY by the subroutine OUTDAT. If he wants to write/store <u>all</u> N available matrices he may

also type the reply 'ALL'. Typing the reply 'ALL' is equivalent to typing all N matrix names. User types desired matrix names format-free. As delimiter the comma ',' is used. Format-free reading of matrix names is performed by subroutine RCHAR, called by either subroutine OUTTTY, OUTLPT and OUTDSK. Typing the reply 'N' causes exit from the subroutine i.e. return to the subroutine OUTDAT. If among the matrix names the user types a nonexisting name, say 'XAX', each subroutine will type the message

MATRIX WITH NAME ⟨XAX⟩ DOES NOT EXIST - TRY AGAIN

followed by the message (8.10), waiting for correct names to be typed. After selecting the desired matrix names, each subroutine OUTTTY, OUTLPT and OUTDSK calls several times its own subroutine by which writing/storing or each particular matrix is done.

Subroutine	OUTTTY	calls	IOMT
"	OUTLPT	"	PRINT
"	OUTDSK	"	DSKWR

The number of calling of these subroutines corresponds to the number of correct matrix names typed as a reply to the message (8.10). After writing/storing the desired number of matrices each subroutine OUT*** returns control to the calling subroutine i.e. to the subroutine OUTDAT, which again types the message (8.9).

8.3.2. Subroutine DSKWR

The subroutine stores on DSK matrix dimensions N, M and all matrix elements of a given N X M matrix. These values are stored on a separate disk record in the user's disk area. The number of matrix elements is limited to 100.

Calling sequence

```
CALL DSKWR(A,N,M,ITA)
```

Description of arguments

A = N X M matrix to be stored on DSK.

N = integer = number of rows in A.

M = integer = number of columns in A.

ITA = up to 3-character ASCII variable specifying matrix name. Usually, ITA is an element of the array IT (see subroutines OUTDAT and OUTDSK).

In order to be able to read from disk a previously stored matrix, all matrices are stored on a separate disk record within the user's data file named MATRIC.DAT. In addition to a matrix name associated with the variable ITA, a record name is chosen by the user in a conversational mode within the subroutine DSKWR. The procedure of selecting a record name and storing a matrix on a disk is explained by the following example.

Assume that the user wants to store on DSK a matrix which has the name A11. The execution of the subroutine DSKWR starts by typing on TTY the message

```
ENTER RECORD NAME FOR MATRIX <A11>                    (8.11)
```

As a reply to the message (8.11) the user may select either an already existing record name, or he may select some new record name. In the first case, say the user in reply to the message (8.11) types a name 'A11R1', the subroutine DSKWR will find out that a matrix under the disk name 'A11R1' already exists on DSK, and will type on TTY the message

```
RECORD WITH NAME ⟨A11R1⟩ ALREADY EXISTS
 TYPE ⟨N⟩,⟨O⟩,⟨S⟩ OR HELP
```

User's reply 'H', stands for Help, causes typing on the TTY the message

```
TO ENTER NEW NAME, OVERWRITE OLD MATRIX OR SKIP
 TYPE ⟨N⟩,⟨O⟩,⟨S⟩ OR HELP
```

The reply 'O' will cause that new data, i.e. dimensions N, M and elements of the new matrix, to be written on the existing disk record under the name A11R1, overwritting old data. If the user wants to save the old data for some future use, he has the opportunity to type the reply 'N. In this case the subroutine DSKWR will type the message

```
TYPE NEW NAME
```

and the user has to type now a new nonexisting disk name. After successful storing of data, for the user's information, the subroutine DSKWR types the message

MATRIX ⟨A11R1⟩ stored in REC # n_1

where n_1 indicates the record number where the matrix has been stored. The reply 'S' causes return to the calling subroutine, i.e. to the subroutine

OUTDSK. Changing and deleting record names may be done by a separate package called EDITD, see Sec. 6.

Subroutine PRINT

The subroutine PRINT is used for printing N X M matrix A on LPT. Before printing matrix elements, the subroutine prints the matrix name, which appears as subroutine argument.

Calling sequence

```
CALL PRINT(A,N,M,IND,NT,IT)
```

The definition of arguments is the same as in the subroutine IOMT, with the exception that the identifier IND may have only two values.

```
For IND = 1 matrix A is printed by format 10F6.2
 "  IND = 2    "   "        "     "   "    5E12.4
```

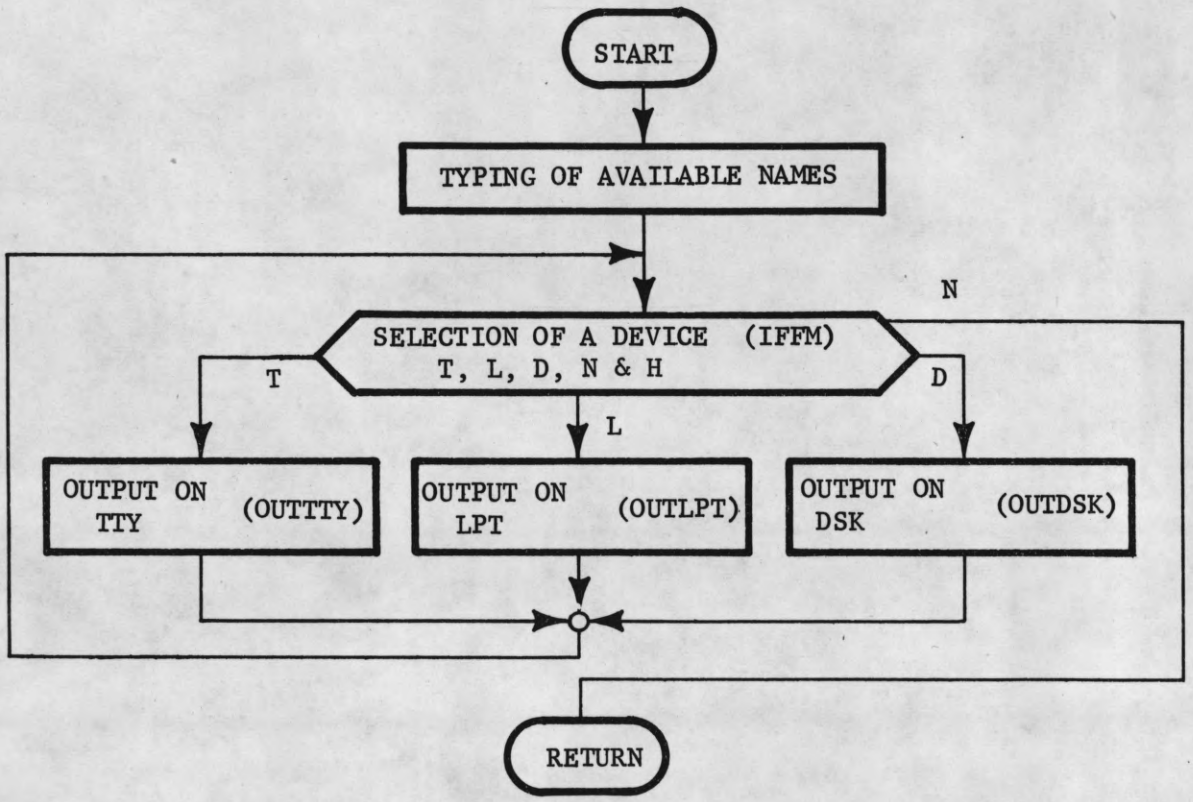

START
TYPING OF AVAILABLE NAMES
SELECTION OF A DEVICE (IFFM)
T, L, D, N & H
N
T
L
D
OUTPUT ON TTY (OUTTTY)
OUTPUT ON LPT (OUTLPT)
OUTPUT ON DSK (OUTDSK)
RETURN
FP-4477

Figure 11

8.4. Subroutine CALDIM

The subroutine CALDIM is used for defining the array M specifying the dimensions of matrices A1,... A6 to be read from TTY or DSK into the core memory, or to be written from core memory into either TTY, LOP or DSK (see subroutines INPDAT and OUTDAT, Secs. 8.2 and 8.3).

Calling sequence

```
CALL CALDIM(J,M,N,I1,I2,I3,I4)
```

Definition of arguments

M = N dimensional integer array to be defined by the subroutine.

J = N dimensional integer array, $1 \leq J_i \leq 4$, $i \in [1,N]$, containing codes for subsequent definition of the array M.

N = number of elements in the arrays M and J.

I1,... I4 = integer variables or constants to be stored into a particular location of the array M.

The particular element M_k, $k \in [1,n]$ of the array M is given by $I(J_k)$, or; for $J_k = \ell$, for

$$I_k = I_\ell , \quad \ell \in [1,4].$$

The array J is usually defined by the FORTRAN DATA statement. Example: For the array J defined by

```
DATA J/1,1,1,2,3,1,1,4,4,1),
```

the statement

```
CALL CALDIM(J,M,8,N,L,K,1)
```

will define the array M in the following way

M(1)=M(2)=M(3)=M(6)=M(8)=N
M(4)=L
M(5)=K
M(7)=1

In other words

M_1	and	M_2	specify	N	X N	matrix
M_3	"	M_4	"	N	X L	"
M_5	"	M_6	"	K	X N	"
M_7	"	M_8	"	1	X N	"

Subroutine TYPDIM

The subroutine TYPDIM types on TTY dimensions of used matrices together with their respective names.

Calling sequence

```
CALL TYPDIM(N,M,IT)
```

Definition of arguments

N = same as in subroutines INPDAT and OUTDAT

M = " " " "

IT = " " " "

The subroutine TYPDIM is particularly useful in the case when the matrices are read from DSK, because it provides a quick check whether the matrices read from DSK are of the appropriate dimensions. An example of this typing is

```
    C       RO      IO
< 3, 4> < 1, 4> < 1, 4>
```

9. INTERROGATION SUBROUTINES

9.1. Subroutine IFF

The subroutine IFF is used in user-computer conversation when only two replies Yes or No are expected.

Calling sequence

CALL IFF ($\$n_1$,$\$n_2$,NT,IT)

Description of arguments

IT = arbitrary text (string of NT alpha-numeric ASCII characters) to be typed on TTY by the subroutine.

NT = integer specifying number of ASCII characters in the text IT, $1 \leq NT \leq 50$.

n_1 = reference number of the statement in the program where the control should be transferred in case the user has typed the reply 'Y'.

n_2 = reference number of the statement where the control should be transferred, in case the user has typed the reply 'N'.

Example:

```
CALL IFF($6,$3,42,'ARE YOU SATISFIED WITH INPUT DATA - Y OR N')
```

In case the user has typed some other character, say 'X', the subroutine will type on TTY the message

? X ? - TRY AGAIN

and in the next row again the text IT will be typed.

9.2. Subroutine IFFM

The subroutine IFFM is used in user-computer conversation when more than two (up to five) replies are possible.

Calling sequence

CALL IFFM($\$n_1$,$\$n_2$,$\$n_3$,$\$n_4$,$\$n_5$,IT1,IT2,IT3,IT4,IT5,NT,IT)

Definition of arguments

IT = same as in the subroutine IFF

NT = same as in the subroutine IFF

IT1,... IT5 = arbitrary one-character ASCII variables.

n_1,... n_5 = reference numbers of the statements in the program where the control should be transferred if the user has typed one of the replies IT1,... IT5.

Example:

```
CALL IFFM($3,$12,$2,$7,$7,1HT,1HL,1HD,1HN,1HN,17,'TYPE T, L, D OR N')
```

Similarly as in the case of subroutine IFF, if user types a reply different from either IT1,... IT5, say X, the subroutine will type on TTY the message

? X ? - TRY AGAIN

and in the next row again the text IT will be typed. In the case when less than 5 five replies are expected, some of the reference numbers n_i and corresponding ASCII variables IT_i should be repeated.

II. COMPUTATIONAL SUBROUTINES

10. LIBRARY LRIC

The library LRIC contains subroutines used exclusively within the package MRIC.

10.1 Subroutine RIC

The subroutine RIC is used for solving the algebraic Riccati matrix equation

$$\begin{aligned} I(K) &\triangleq KA + xA^T K - KSK + Q = 0 \\ S &\triangleq BR^{-1} B^T \end{aligned} \tag{10.1}$$

by the Newton-Lyapunov iteration method [5],[6].

Calling sequence

```
CALL RIC(A,B,Q,R,AK,EPS,N,M,IT,IJ,AF)
```

Definition of arguments

A, B, Q and R = (n × n), (m × n), (n × n) and (m × m) matrices, respectively appearing in (10.1).

AK = (n × n) matrix, initial guess for the Riccati gain K. After the execution of the subroutine the array AK contains the Riccati gain satisfying the following stopping condition

$$\|I(K)\|/\|K\| \leq e \ll 1. \tag{10.2}$$

EPS = sufficiently small positive number corresponding to e.

N and M = integers corresponding to n and m, respectively.

IT = maximal number of iterations.

IJ = integer specifying number of iterations used in satisfying the stopping condition (10.2).

IJ > IT designates that within IT iterations no satisfactory solution has been found.

In the case when the pair {A,B} is not stabilizable, the subroutine types the message

SYSTEM IS NOT STABILIZABLE

sets IJ = 0 and returns to the calling program. AF = (n X n) matrix defined by:

$$AF = A - S*AK.$$

In the case that $\det|R| \leq 10^{-12}$ the subroutine types the message

DET. OF R IS LESS THAN 0.100-11

sets IJ = 0 and returns to the calling program.

10.2. Subroutine LYAP

The subroutine LYAP uses the Smith method [7], for solving the linear Lyapunov matrix equations of the forms

$$L_o(x) = A^T X + XA + B = 0 \tag{10.3}$$

or

$$L_1(x) = AX + XA^T + B = 0 \tag{10.4}$$

Calling sequence

```
CALL LYAP(A,B,X,Q,EPS,N,IT,ITRANS)
```

Definition of arguments

A and B = given (n X n) matrices. All eigenvalues of the matrix A should have negative real parts.

X = (n X n) matrix containing solution satisfying either

$$\|L_o(X)\|/\|X\| \leq e \ll 1, \quad \text{or} \quad \|L_1(X)\|/\|X\| \leq e \ll 1. \tag{10.5}$$

Q = positive scalar, usually Q = 1.

EPS = sufficiently small positive number corresponding to e.

N = integer corresponding to n.

IT = integer variable containing maximal number of iterations.

ITRANS = identifier. For ITRANS = 0 equation (10.3) is solved, while for ITRANS = 1 equation (10.4) is solved.

If within IT iterations no satisfactory solution X satisfying (10.5) is f found, the subroutine types the message

```
LYAP NO. OF ITER= xxxx
```

where xxxx correspond to the value of IT.

10.3. Subroutine TRC

The subroutine TRC calculates the error matrix

$$Y(K) = KA + A^{T}K - KSK + Q$$

used within the subroutine RIC.

Calling sequence

```
CALL TRC(A,S,Q,AK,N,Y,C,D)
```

Definition of arguments

A,S,Q,AK = given (n X n) matrices. AK represents the matrix K.

N = integer corresponding to n.

Y = (n X n) matrix containing the error matrix Y(K).

C,D = working (n X n) matrices.

Subroutine STT

The subroutine STT checks the stability of a given (n X n) matrix A.

Calling sequence

```
CALL STT(A,N,RR,RI,IND)
```

Definition of arguments

A = given (n X n) matrix,

N = integer specifying the order n,

RR and RI = n dimensional arrays containing real and imaginary parts of the eigenvalues of the given matrix A,

IND = stability identifier. If the matrix A is stable, the subroutine sets IND = 1, otherwise it sets IND = 0.

Subroutine TEST

The subroutine TEST checks the difference between two given (n X n) matrices.

Calling sequence

```
CALL TEST(AK,X,N,EPS,P1,IEK)
```

Definition of arguments

AK and X = given (n X n) matrices,

N = integer corresponding to n,

EPS = sufficiently small positive number corresponding to e,

P1 = working (n X n) matrix,

IEK = identifier.

If $\|AK - X\|/\|AK\| \leq e$, the subroutine sets IEK = 1, otherwise it sets IEK = 0. Before returning to the calling program the subroutine sets AK = X.

10.4. Subroutines STABK and JACC

The subroutines STABK and JACC are used for providing the symmetric matrix K which stabilizes the closed-loop matrix A - SK. The detailed description of the algorithm is given in [6].

Background:

Given (n X n) matrices A and S and positive scalars c and e, where A is unstable and S positive semidefinite, the subroutine STABK calculates positive definite matrix S* given by S* = S + eI, I being (n X n) unity matrix.

The subroutine JACC calculates a symmetric matrix K satisfying the condition that the matrix A_f = A - S*K has all eigenvalues equal to -c.

Calling sequence for the subroutine STABK

```
CALL STABK(A,B,AK,S,N,CC,ES,C,IND)
```

Definition of arguments

A and S = given (n X n) matrices.

N = integer corresponding to n.

CC and ES = positive scalars corresponding to c and e, respectively.

B,AK,C = (n X n) matrices corresponding to S*, K and A_f, respectively.

IND = stability identifier. If matrix A_f is stable the subroutine sets IND = 1, otherwise it sets IND = 0.

Calling sequence for the subroutine JACC

```
CALL JACC(A,S,N,AK,C)
```

Definition of arguments

A and S = given (n X n) matrices corresponding to A and S^*, respectively,

N = integer corresponding to n,

AK = (n X n) matrix corresponding to K, calculated by the subroutine,

C = positive scalar corresponding to c.

11. LIBRARY LPPL

The library LPPL contains subroutine used exclusively within the package MPPL.

11.1. Subroutine PPL

The subroutine PPL is used in pole placement and observer design.

11.1.1. Pole placement algorithm

1. Given $\{A,B\}$, where rank $|B| = m$, the controllability matrix Q_c and controllability indices n_i, $i \in [1,m]$, are calculated. Q_c and n_i are calculated using the subroutine COIN2. The structure of Q_c is

$$Q_c = |\underbrace{B}_{\eta_1} : \underbrace{\widetilde{AB}}_{\eta_2} : \underbrace{\widetilde{A^2B}}_{\eta_3} : \cdots : \underbrace{\widetilde{A^{\nu-1}B}}_{\eta_\nu}| \qquad (11.1)$$

$$0 < \eta_\nu \leq \eta_{\nu-1} \leq \cdots \leq \eta_3 \leq \eta_2 \leq \eta_1 = m$$

$$\sum_{i=1}^{\nu} \eta_i = n.$$

Submatrices $\widetilde{A^{i-1}B}$ are $n \times \eta_i$ matrices containing columns of matrices A^iB which are linearly independent of all columns of the matrix

$$Q_c^i = |B : \widetilde{AB} : \cdots : \widetilde{A^{i-1}B}|.$$

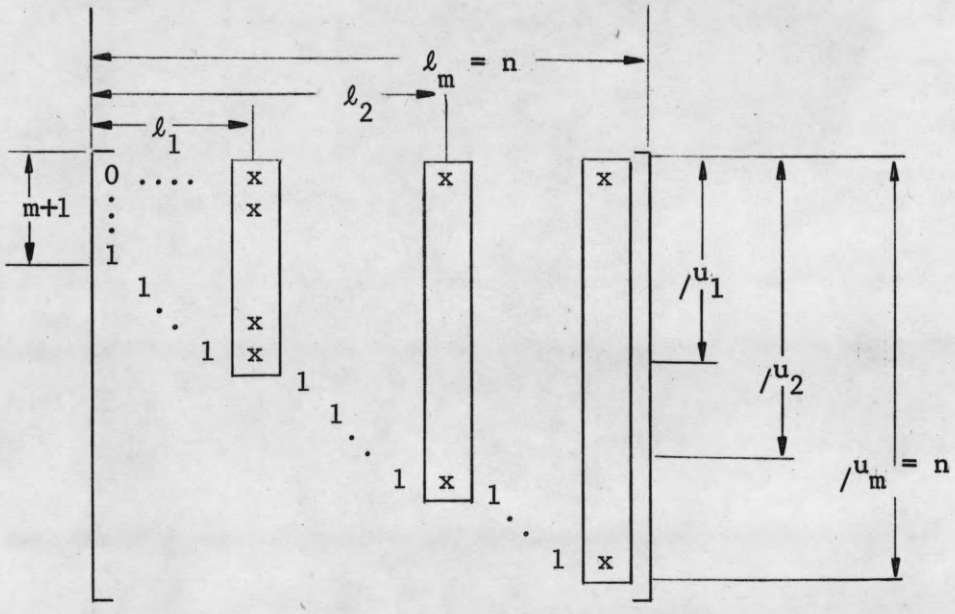
Integer η_i denotes number of columns in the matrix $\widetilde{A^{i-1}B}$. Controllability indices n_i are defined as the number of appearances of a particular column b_i in the matrix Q_c. Thus

$$\sum_{i=1}^{m} n_i = n. \qquad (11.2)$$

2. Applying the state transformation

$$A_1 = Q_c^{-1} A Q_c$$
$$B_1 = Q_c^{-1} B = \left| \begin{matrix} \cdots \\ I \\ \cdots \\ 0 \end{matrix} \right| \qquad (11.3)$$

a new realization of the open-loop system is obtained. Matrix A_1 has a form

$A_1 =$ 

(11.4)

The meaning of ℓ_i and $/u_i$ is explained by (11.4). The values ℓ_i and $/u_i$ are uniquely determined by the values of n_i, $i \in [1,m]$.

3. Using $(m \times n)$ matrix C_1 defined by

$$C = \begin{bmatrix} 0\ 0 \ldots & \ldots\ 1 \\ \vdots & \\ 0\ 0 \ldots\ldots 1 & \\ 0\ 0 \ldots 1 & \end{bmatrix} \qquad (11.5)$$

where $c_{ij} = 1$, $i \in [1,m]$ only for $j = \ell_{m-i+1}$, another transformation matrix Q_o is obtained

$$Q_o = \begin{bmatrix} C_1 & \eta_1 \\ \hline \widetilde{C_1 A} & \eta_2 \\ \hline \vdots & \\ \hline \widetilde{C_1 A^{\nu-1}} & \eta_\nu \end{bmatrix} \tag{11.6}$$

$\widetilde{C_1 A^i}$ is a $\eta_i \times n$ matrix containing the first η_i rows of the matrix $C_1 A^i$, η_i being defined by (11.1). From (11.6) and according to the structure of the pair $\{A_1, C_1\}$ it may be concluded that

$$\rho|Q_o| = n.$$

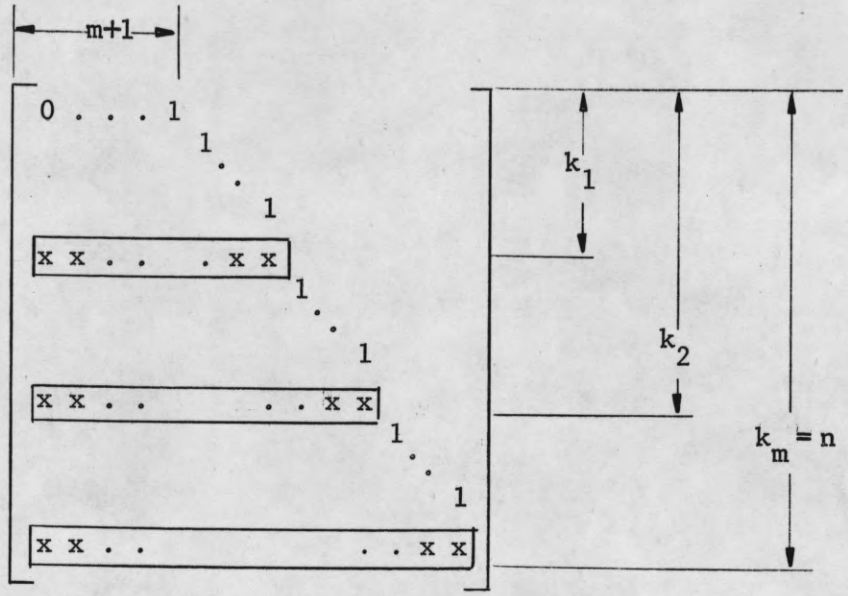
Using Q_o as another transformation matrix a transformed pair

$$A_2 = Q_o A_1 Q_o^{-1}$$

$$B_2 = Q_o B_1$$

is obtained. The matrix A_2 is of the form

$A_2 =$



(11.7)

Integers k_i, $i \in [1,m]$, indicate locations of rows of the matrix A_2 with nonzero and nonunity elements. In matrix B_2 only rows b_{k_i} contain nonzero elements.

4. Using the obtained pair $\{A_2, B_2\}$ the pole placement condition may be expressed by

$$A_2 + B_2 F_2 = A_{2d} \tag{11.8}$$

where the eigenvalues of the matrix A_{2d} should correspond to the desired eigenvalues of the closed-loop system. From (11.8) and according to the form of the pair $\{A_2, B_2\}$ it may be concluded that the matrix A_{2d} has the same structure as the matrix A_2. Thus, the pole placement condition becomes

$$B_2 F_2 = A_{2d} - A_{21} \triangleq D_2 \tag{11.9}$$

where the matrix D_2, similarly as B_2, has nonzero elements only in rows whose indices correspond to values k_i, $i \in [1,m]$. Introducing the definitions

$$B_2^* \triangleq \begin{vmatrix} b_{k_1} \\ \vdots \\ b_{k_m} \end{vmatrix} \qquad D_2^* \triangleq \begin{vmatrix} d_{k1} \\ \vdots \\ d_{km} \end{vmatrix} \tag{11.10}$$

where matrices B^* and D^* contain nonzero rows from matrices B_2 and D_2, respectively, condition (11.9) becomes

$$B^* F_2 = D^*.$$

Since m X m matrix B^* is always of the triangular form, with unities in the main diagonal, the feedback gain F_2 becomes

$$F_2 = B^{*-1} D^*.$$

Returning back to the original realization, the feedback gain F, which assures the desired eigenvalues of the closed-loop system

$$A + BF = A_f$$

is given by

$$F = F_2 Q_o Q_c^{-1} = B^{*-1} D^* Q_o Q_c^{-1}.$$

Calculation of the transformed pairs $\{A_1, B_1\}$ and $\{A_2, B_2\}$ is done by the subroutine TRANS. Given the desired eigenvalues the subroutine FAD calculates the nonzero and nonunity rows of the matrix A_{2d}. Each row in A_{2d} contains the coefficients of a characteristic polynomial corresponding to a subset of n_i desired eigenvalues, n_i being the controllability index. Given n_i eigenvalues, coefficients of the corresponding characteristic equation are calculated by the subroutine EGVCHE. Reduction of B_2 and D_2 to B^* and D^* is done by the subroutine RED. Calculation of the matrix Q_o is done by the

subroutine FRT. Observer design is done by the same algorithm, treating the pair $\{A^T, C^T\}$ and calculating the $(k \times n)$ matrix K^T satisfying that the matrix $A_{of}^T = A^T + C^T K^T$ has specified eigenvalues.

11.1.2. Calling sequence

```
CALL PPL(A,B,RR,RI,AD,N,M,EPS,FR,P1,AF,ICO,IND)
```

Definition of arguments

A = $(n \times n)$ given open-loop system matrix,

IND = identifier $\begin{cases} \text{for pole placement IND = 'C'} \\ \text{for observer design IND = 'O'} \end{cases}$,

B = $\begin{cases} \text{for IND = 'C' } (n \times m) \text{ input matrix B} \\ \text{for IND = 'O' } (k \times n) \text{ output matrix C} \end{cases}$,

RR and RI = n dimensional arrays containing respectively real and imaginary parts of desired closed-loop eigenvalues,

AD = working $(n \times n)$ matrix,

N = integer corresponding to the system order n,

M = integer $\begin{cases} \text{for IND = 'C' corresponding to m} \\ \text{for IND = 'O' corresponding to k} \end{cases}$,

FR,P1 &AF = $(n \times n)$ matrices. For IND = 'C' these matrices correspond to feedback gain matrix F, product -BF, and closed-loop system matrix $A_f = A + BF$, respectively, while for IND = 'O', these matrices correspond to observer gain K, product -KC, and closed-loop observer matrix $A_{of} = A + KC$, respectively.

EPS = sufficiently small positive scalar,

ICO = identifier. If either the pair $\{A,B\}$ is not controllable, or the pair $\{A,C\}$ is not observable the subroutine sets ICO = 1, types the message,

SYSTEM IS NOT CONTROLLABLE/OBSERVABLE

and returns to the calling program.

If rank $|B| < m$ or rank $|C| < k$, the subroutine sets ICO = 1, types the message

MATRIX B/C IS NOT OF FULL RANK

and returns to the calling program. Otherwise it sets ICO = 0. The pole placement algorithm assumes that rank $|B| = m$, i.e. that there are no redundant inputs. Prior to the use of the subroutine PPL, user should eliminate all existing redundant inputs.

11.2. Subroutine COIN2

Given (n X n) system matrix A, and either (n X m) input matrix B or (k X n) output matrix C, the subroutine COIN2 calculates either the controllability indices and the controllability matrix, or the observability indices and the observability matrix. The subroutine also checks rank of the matrix B (or C).

Calling sequence

```
CALL COIN2(A,B,N,M,NZ,NV,LV,IND,EPS,H)
```

The definition of arguments will be given in the case of calculating the controllability indices n_i and controllability matrix Q_c defined by (11.1) and (11.2),

A and B = (n X n) and (n X m) matrices specifying the pair {A,B},

N and M = integers corresponding to n and m, respectively,

NZ = m dimensional array containing controllability indices n_i

NV = m+1 dimensional array containing ordered controllability indices, $NV(1) \leq NV(2) \leq \cdots \leq NV(m)$. NV(m+1) contains sums of all controllability indices n_i, $i \in [1,m]$,

LV = m dimensional array containing values ℓ_i defined by (11.4),

IND = identifier $\left\{\begin{array}{l}\text{IND = 'C' for calculating the controllability matrix}\\ \text{IND = 'O' for calculating the observability matrix}\end{array}\right\}$,

EPS = sufficiently small positive number used in creating controllability matrix Q_c,

H = (n X n) controllability matrix calculated by the subroutine.

If the pair {A,B} is not controllable, or rank|B| < m, the subroutine types on TTY either

SYSTEM IS NOT CONTROLLABLE/OBSERVABLE

or

MATRIX B/C IS NOT OF FULL RANK

and returns to the calling program.

11.3. Subroutine RED

The subroutine RED is used for reducing (n X n) matrix D_2 and (n X m) matrix B_2 to (m X n) matrix D_2^* and (m X m) matrix B_2^*, defined by (11.10), respectively.

Calling sequence

```
CALL RED(A,N,M,K,LV,B)
```

Definition of arguments

A = given (n X k) matrix corresponding to either D_2 or B_2,

LV = m dimensional integer array indicating nonzero rows in the matrix A,

N and M = integers corresponding to n and m, respectively

K = integer corresponding to either n or m depending on whether A corresponds to D_2 or B_2,

B = resulting (m X k) matrix containing nonzero rows of the matrix A.

Subroutine FAD

Given desired eigenvalues, the subroutine FAD calculates the nonzero and nonunity rows of the matrix A_{2d}, defined by (11.8). Each row in A_{2d} contains the coefficients of a characteristic polynomial corresponding to a subset of n_i desired eigenvalues, n_i being controllability index.

Calling sequence

```
CALL FAD(NV,M,N,L1,RR,RI,P2,P1)
```

Definition of arguments

NV = m dimensional integer array containing ordered controllability indices, see subroutine PPL,

N,M = integers corresponding to n,m,

L1 = working m dimensional integer array,

P2 = (m X n) working matrix,

P1 = (m X n) matrix containing m nonzero and nonunity rows of the matrix A_{2d}.

Subroutine EGVCHE

Given n dimensional arrays containing real and imaginary parts of n eigenvalues, the subroutine EGVCHE calculates coefficients of the corresponding characteristic polynomial.

Calling sequence

```
CALL EGVCHE(RR,RI,N,Z,IZ)
```

Definition of arguments

RR, RI = n dimensional arrays specifying real and imaginary parts of n given eigenvalues.

N = integer corresponding to n,

Z = (n+1) dimensional array containing coefficients z_i of the polynomial $z(s) = \sum_{i=1}^{n} z_i s^{i-1} + s^n$. The value of z_{n+1} is always equal to unity, $z_{n+1} = 1$.

Subroutine FRT

Given matrix A_1 and m dimensional array containing integers ℓ_i, the subroutine FRT calculates the matrices C_1 and Q_o, defined by (11.5) and (11.6).

Calling sequence

```
CALL FRT(A,LV,NV,NVM,N,M,P1,P2,Q)
```

Definition of arguments

A = given (n × n) matrix,

LV = m dimensional array containing integers ℓ_i,

NV = m dimensional integer array containing ordered controllability indices n_i,

NVM = integer, equal to NV(m),

N,M = integers corresponding to n,m,

P1 = resulting (n × n) matrix corresponding to the matrix Q_o,

P2,Q = working (n × n) matrices.

12. LIBRARY LGEN

The library LGEN contains general subroutines used by all packages.

12.1. Subroutine RANKR

Given a realization $\{A,B,C\}$ and all eigenvalues $\lambda_i = \delta_i \pm j\omega_i$ of the matrix A, the subroutine checks either the stabilizability of $\{A,B\}$ or the detectability of $\{A,C\}$. The subroutine requires a small positive scalar e.

Background

The stabilizability of a given pair $\{A,B\}$ is checked by investigating the rank of a $(n \times n+m)$ complex matrix [8] given by

$$Q_c = |I\lambda_i - A \vdots B|. \tag{12.1}$$

It may be shown that in the case of complex eigenvalue λ_i, the rank of (12.1) may be checked by calculating the rank of a real $(2n \times 2n+2m)$ matrix given by

$$Q_{cr} = \left|\begin{array}{c|c|c|c} I\delta_i + A & B & -I\omega_i & 0 \\ \hline I\omega_i & 0 & I\delta_i + A & B \end{array}\right| \tag{12.2}$$

Calling sequence

```
CALL RANKR(A,B,RR,RI,N,M,EPS,IR,IND)
```

Definition of arguments

A = given $(n \times n)$ matrix,

RR and RI = given n dimensional arrays containing real and imaginary parts, δ_i and ω_i, respectively of λ_i,

EPS = given small positive number corresponding to e,

IND = identifier { IND = 'C' for checking stabilizability ; IND = 'O' for checking detectability }

B = { for IND = 'C' given (n × m) input matrix B ; for IND = 'O' given (k × n) output matrix C }

M = integer { for IND = 'C' corresponding to m ; for IND = 'O' corresponding to k }

IR = rank identifier

for IND = 'C' and {A,B} being not controllable, IR = 0,

for IND = 'O' and {A,C} being not observable, IR = 0,

otherwise IR ≠ 0.

In the case of IR = 0 the subroutine types on TTY either

MODE xxxxxxx +/- J yyyyyyy not controllable

or

MODE xxxxxxx +/- J yyyyyyy not observable

where xxxxxxx and yyyyyyy represent respectively real and imaginary parts of the eigenvalue being either not controllable or not observable. The required matrix Q_{cr}, given by (12.2), is calculated by the subroutine JOIN.

Subroutine JFORM

Given the realization {A,B,C} the subroutine JFORM calculates the modal transformation matrix T, its inverse T^{-1} and the transformed triple $\{\tilde{A},\tilde{B},\tilde{C}\}$ given by

$$\tilde{A} = T^{-1} A T$$
$$\tilde{B} = T^{-1} B \qquad (12.3)$$
$$\tilde{C} = C T$$

where the matrix $\tilde{A}$ is in Jordan form. The application of the subroutine is limited to cyclic matrices A [11].

Background

The transformation matrix T is calculated by

$$T = T_1 T_2$$

where T_1 transforms a given matrix A into the observable companion form [9], while T_2 contains the eigenvectors of the matrix $T_1^{-1} A T_1$, [10]. Cyclicity of the matrix A is checked calculating the determinant of the transformation matrix T_1. Calculation of the matrix T_1 requires an arbitrary n dimensional column vector generated within the subroutine using the random number generator RAN.

Calling sequence

```
CALL JFORM(A,B,C,N,M,K,T,TI)
```

Definition of arguments

A, B and C = given (n × n), (n × m) and (k × n) matrices corresponding to the given realization {A,B,C}. After execution the transformed triple $\{\tilde{A},\tilde{B},\tilde{C}\}$ appears in the same locations.

N, M and K = integers corresponding to n, m and k respectively.

T and TI = (n × n) matrices corresponding to transformation matrices T and T^{-1}, respectively.

Calculation of the matrix T_1 is done using the subroutine COMF, while the matrix T_2 is obtained by the subroutines CHEQ and EGT. Eigenvalue sorting is done by the subroutine ORD. Calculation of the transformed triple is done by the subroutine TRANS.

12.2. Subroutine COMF

Background

Given (n × n) matrix A and n dimensional column vector r the subroutine calculates a (n × n) matrix C given either by

$$C = T_c = |A^{n-1}r \vdots \cdots a \vdots A^2 r \vdots A r \vdots r| \qquad (12.4)$$

or

$$C = T_o = \begin{vmatrix} r^T \\ \hline r^T A \\ \hline r^T A^2 \\ \hline \vdots \\ \hline r^T A^{n-1} \end{vmatrix}$$

It is well known that if det $T_c \neq 0$ and det $T_o \neq 0$ the transformations

$$A_c = T_c^{-1} A T_c \quad \text{and} \quad A_o = T_o A T_o^{-1}$$

will bring matrices A_c and A_o in controllable and observable companion forms, respectively.

Calling sequence

```
CALL COMF(A,R,N,IND,C,T)
```

Definition of arguments

A = given (n × n) matrix,

R = n dimensional vector,

N = integer corresponding to n,

IND = identifier = $\left\{\begin{matrix}'C'\\'O'\end{matrix}\right\}$,

C = resulting (n X n) matrix; for IND = 'C' C = T_c while for IND = 'O' C = T_o,

T = working (n X n) matrix.

Subroutine EGT

Given (n X n) matrix A in observable companion form [9], real parts δ_i and imaginary parts ω_i of eigenvalues λ_i of A, the subroutine EGT calculates the eigenvector matrix P. The recurrent expressions used in calculating the columns of the matrix P are given in [10].

Calling sequence

```
CALL EGT(A,RR,RI,N,P)
```

Definition of arguments

A = given (n X n) matrix in observable companion form,

RR and RI = n dimensional array containing real and imaginary parts of eigenvalues λ_i of A,

N = integer corresponding to the matrix order n,

P = resulting (n X n) matrix.

12.3. Subroutine TRANS

Given the triple $\{A,B,C\}$ and either one or both matrices T and T^{-1}, the subroutine calculates the transformed triple

$$\begin{aligned} \tilde{A} &= T^{-1} A T \\ \tilde{B} &= T^{-1} B \\ \tilde{C} &= C T \end{aligned} \qquad (12.5)$$

Calling sequence

```
CALL TRANS(A,B,C,T,TI,N,M,K,IND)
```

Definition of arguments

A, B and C = given (n × n), (n × m) and (k × n) matrices corresponding to the given realization $\{A,B,C\}$. After execution of the subroutine the transformed triple (12.5) appears in the same locations.

T and TI = (n × n) matrices corresponding to T and T^{-1}, respectively.

N, M and K = integers corresponding to n, m and k, respectively.

IND = identifier $\begin{cases} \text{IND = 0, both T and } T^{-1} \text{ are given} \\ \text{IND = 1, only T is given} \\ \text{IND = 2, only } T^{-1} \text{ is given} \end{cases}$.

In the case of IND ≠ 0 matrix T^{-1} or T is calculated within the subroutine.

Subroutine CHEQ

Given (n × n) matrix A the subroutine calculates the coefficients f_i, $i \in [1,n]$ of the characteristic polynomial

$$f(s) = \sum_{i=1}^{n} f_i s^{i-1} + s^n.$$

Calling sequence

```
CALL CHEQ(A,N,F,B,C)
```

Definition of arguments

A = given (n × n) matrix,

N = integer corresponding to n,

F = $\{f_i\}$ = n dimensional array containing coefficients f_i,

B and C = working (n × n) matrices.

12.4. Subroutine RANK2

Given (m × k) matrix H the subroutine checks the rank of the matrix H and calculates det H^TH. Rank calculation requires a small positive number e.

Calling sequence

```
CALL RANK2(H,P,D,M,K,IR,EPS)
```

Definition of arguments

H = given (m × k) matrix,

P = working (m × k) matrix,

D = determinant of $[H^TH]$,

M and K = integers corresponding to m and k,

IR = rank identifier; IR = 0 if rank$|H| < \min(m,k)$, otherwise IR = min(m,k),

EPS = scalar corresponding to sufficiently small number e.

Function DET

The function DET is used for determinant calculation of a given (n × n) matrix. This function has been taken from [12].

Calling sequence

```
D = DET(A,N)
```

Definition of arguments

A = given (n × n) matrix,

N = integer corresponding to n.

Subroutine JOIN

Given matrices A_{11}, A_{12}, A_{21} and A_{22} with corresponding dimensions, (see scheme) the subroutine joins them into a composite (n × m) matrix A.

Calling sequence

CALL JOIN(A11,A12,A21,A22,N1,M1,N2,M2,N,M,A)

Definition of arguments

A11,A12,A21,A22 = given matrices of the corresponding dimensions,

N1,M1,N2,M2 = integers specifying dimensions of the given matrices,

N and M = integers corresponding to $n = n_1+n_2$ and $m = m_1+m_2$, respectively,

A = resulting composite matrix.

Subroutine DJOIN

Given (n × m) matrix A and dimensions n_1, m_1, n_2 and m_2 specifying the desired partitioning, see scheme, the subroutine calculates the submatrices A_{11}, A_{12}, A_{21} and A_{22}.

Calling sequence

CALL DJOIN(A11,A12,A21,A22,N1,M1,N2,M2,N,M,A).

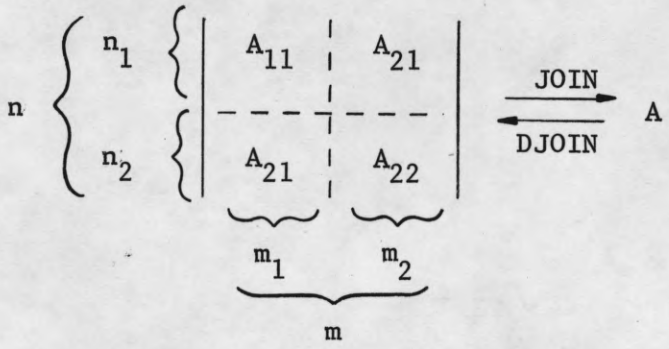
Definition of arguments

A = given (n × m) matrix,

N and M = integers corresponding to n and m, respectively,

N1,M1,N2,M2 = integers specifying the desired partitioning,

A11,A12,A21 and A22 = resulting submatrices of the corresponding dimensions.


n
n1
n2
A11
A21
A21
A22
m1
m2
m
JOIN
DJOIN
A

12.5. Subroutine DREAL

Given time interval Δt and a pair $\{A,B\}$ representing either the continuous or discrete system realization, the subroutine calculates the corresponding discrete or continuous pair $\{F,G\}$, respectively.

Calling sequence

```
CALL DREAL(A,B,N,M,DT,F,G,IND)
```

Description of arguments

A and B = given (n × n) and (n × m) matrices corresponding to the given continuous or discrete pair $\{A,B\}$,

N and M = integers corresponding to n and m, respectively,

DT = scalar corresponding to Δt,

F and G = resulting (n × n) and (n × m) matrices,

IND = identifier = $\begin{Bmatrix} \text{'D'} \\ \text{'C'} \end{Bmatrix}$.

For IND = 'D' the subroutine calculates the discrete pair $\{F,G\}$ from the given continuous $\{A,B\}$, while for IND = 'C' the continuous pair $\{F,G\}$ is calculated from the given discrete pair $\{A,B\}$. In other words, for IND = 'D' matrix F satisfies $F = \exp(A\,\Delta t)$ while for IND = 'C' the relation between calculated F and the given A is inverse, i.e. $A = \exp(F\,\Delta t)$. In both cases the resulting matrix F is obtained using the method for calculating the matrix polynomials described in [13].

Subroutine MTF

Given a realization $\{A,B,C\}$, the subroutine calculates coefficients f_i, $i \in [1,n]$ of the characteristic polynomial

$f(s) = \det(Is - A) = s^n + \sum_{i=1}^{n} f_i s^{i-1}$ and either one or both coefficients w_{gj}^i and t_{gj}^i defining state and output transfer function matrices, respectively,

$$(Is - A)^{-1}B = W(s)/f(s); \qquad W(s) = \{w_{hj}(s)\}$$

$$C(Is - A)^{-1}B = T(s)/f(s); \qquad T(s) = \{t_{gj}(s)\}$$

$$w_{hj}(s) = \sum_{i=1}^{n} w_{hj}^i s^{i-1}; \qquad h \in [1,n]; \quad j \in [1,m]$$

$$t_{gj}(s) = \sum_{i=1}^{n} t_{gj}^i s^{i-1}; \qquad g \in [1,k]; \quad j \in [1,m].$$

Calling sequence

```
CALL MTF(A,B,C,N,M,K,IND,F,W,T)
```

Definition of arguments

A, B and C = given (n × n), (n × m) and (k × n) matrices, respectively,

N, M and K = integers corresponding to n, m and k,

F = n dimensional array containing coefficients f_i,

W = nnm dimensional array containing coefficients

w_{hj}^i, $i \in [1,n]$; $h \in [1,n]$; $j \in [1,m]$,

T = nkm dimensional array containing coefficients

t_{gj}^i, $i \in [1,n]$; $g \in [1,k]$; $j \in [1,m]$,

IND = identifier.

For IND = 1 the subroutine calculates both arrays W and T, while for IND = 0 and IND = -1 only the array T and the array W is calculated, respectively. In the cases of IND = 0 and IND = -1, it is not necessary to dimension the noncalculated array. A dummy scalar variable may be used instead. The method used for calculation of the coefficients of W(s) and T(s) is presented in [14].

13. LIBRARY LOUT

The library LOUT contains subroutines used in conjunction with the plotting subroutines PLXU and EGVP, described in Sec. 7.

13.1. Subroutine CALYU

Given $(k \times n)$ matrix $U = \{u_{ij}\}$, $i \in [1,k]$, $j \in [1,n]$, $u_{ij} = x_j(t_i)$ containing responses of the state vector $x(t)$ at time instances t_i, the subroutine CALYU calculates the corresponding values of either control vector $u(t)$ or output vector $y(t)$, see subroutine PLXU and block diagram, Fig. 2.

Calling sequence

```
CALL CALYU($n1,B,R,AK,Z,U,V,N,M,K,IO,IND)
```

Definition of arguments

n_I = statement number in the calling program (subroutine PLXU), where the control is to be transferred.

B,R,AK = matrices used in calculation of either u(t) or y(t), see following table.

Z = $(n \times n)$ working matrix.

V = $(k \times n)$ working matrix.

N = integer corresponding to n.

M = integer corresponding to dimension of either u(t) or y(t), see following table.

K = integer corresponding to k

IND = identifier = $\begin{Bmatrix} \text{'RC'} \\ \text{'PP'} \\ \text{'PO'} \\ \text{'AN'} \end{Bmatrix}$

$$IO = \text{second identifier} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}.$$

The use of identifiers IND, IO, matrices B, R, AK and integer M in calculating either u(t) or y(t) is explained in the following table

	Package	IND	IO	Calculation done
(a)	MRIC	'RC'	1	$u(t) = -R^{-1} B^T Kx(t)$
(b)	MPPL	'PP'	1	$u(t) = Fx(t)$
(c)	MPPL	'PO'	1	$u(t) = \left\| F \vdots -F \right\| \left\| \begin{matrix} x(t) \\ e(t) \end{matrix} \right\|$
(d)	MANAL	'AN'	1	$y(t) = Cx(t)$
(e)	MANAL	'AN'	2	$u(t) = Fy(t)$

Matrices R^{-1}, B and K in (a) correspond to matrices R, B and AK in the calling sequence. Matrix F in (b) and (c) corresponds to the matrix B in the calling sequence. Matrix C in (d) corresponds to the matrix AK in the calling sequence. Matrix F in (e) corresponds to the matrix R in the calling sequence. Only in (d) M corresponds to the dimension of y(t), otherwise M corresponds to the dimension of u(t).

13.2. Subroutine MRLOC

The subroutine performs selection of a part of the s-plane to be plotted on the terminal screen, see flowchart of the plotting subroutine EGVP.

Calling sequence

```
CALL MRLOC(N,RR,RI,X,Y)
```

Definition of arguments

RR and RI = n dimensional arrays containing respectively real and imaginary parts of given eigenvalues.

N = integer corresponding to n.

X and Y = working n dimensional arrays.

Subroutine RLOC

The subroutine plots a part of the s-plane containing eigenvalue locations.

Calling sequence

```
CALL RLOC(N,RR,RI,X,Y)
```

Definition of arguments

Same as in subroutine MRLOC.

Subroutine PLOTN

Given k dimensional array $x = \{x_i\}$, $i \in [1,k]$ and a $(k \times n)$ matrix $Y = \{y_{ij}\}$, $y_{ij} = y_j(x_i)$, the subroutine plots on the terminal screen n curves, $j \in [1,n]$, defined by $y_j(x_i)$. The scales for individual y_j are

either different or common. The selection of a particular option is done in the conversational mode, see flowchart of the plotting subroutine PLXU.

Calling sequence

```
CALL PLOTN(N,K,X,Y)
```

Definition of arguments

N and K = integers corresponding to n and k, respectively,

X = given k dimensional array, $X = \{x_i\}$, $i \in [1,k]$,

Y = given (k X n) matrix, $Y = \{y_{ij}\}$, $y_{ij} = y_j(x_i)$, $j \in [1,n]$.

13.3. Subroutine RESP

Given regular ($n \times n$) matrix A, n dimensional initial condition vector x^o, ($n \times n_d$) disturbance matrix $E = |e^1 \vdots e^2 \vdots \cdots \vdots e^{n_d}|$, time increment Δt, and number of intervals k, the subroutine calculates responses $x(t_i)$, $t \in [1,k]$, where the vector $x(t)$ is defined by

$$\dot{x}(t) = Ax(t) + Ez(t) \quad , x(0) = x^o. \tag{13.1}$$

Elements of the n_d dimensional column vector

$$z^T(t) \triangleq |1 \vdots t \vdots t^2 \vdots \cdots \vdots t^{n_d-1}|$$

are step, ramp, etc.

The algorithm implemented in the subroutine RESP is based on calculating $x(t)$ as a sum of

$$x(t) = z(t) + y(t) \tag{13.2}$$

where n dimensional vectors $z(t)$ and $y(t)$ are defined by

$$\begin{aligned} \dot{z}(t) &= Az(t); \quad z(0) = x^o + g^1 \\ y(t) &= -\sum_{j=1}^{n_d} g^j \, t^{j-1}/(j-1)! \end{aligned} \tag{13.3}$$

n dimensional vectors g^j, $j \in [1,n_d]$ are related to vectors e^j, columns of the matrix E by

$$\begin{aligned} g^j &= A^{-1} \, [(j-1)! \, e^j + g^{j+1}]; \quad j \in [1,n_d] \\ g^{n_d+1} &= 0. \end{aligned} \tag{13.4}$$

The expressions (13.2) and (13.4) may be proved applying the Laplace transform and substituting in (13.7) $z(s)$ and $y(s)$ obtained from (13.3).

The responses z(t) are calculated by either the subroutine OUT2 or OUT. Normally, the subroutine OUT2, applicable only for cyclic matrices A, is used. If the matrix A is noncyclic, the subroutine RESP types on TTY the message

MATRIX NONCYCLIC

and proceeds by calculating z(t) applying the subroutine OUT. The reason for using both subroutines is that there was no sufficient time for detailed checking the subroutine OUT. If Δt is too large, the subroutine OUT might not work properly.

Calling sequence

```
CALL RESP(A,DT,N,K,XO,XADD,RES,B,M)
```

Definition of arguments

A = given $(n \times n)$ matrix,

DT = scalar corresponding to Δt,

N and K = integers corresponding to n and k,

XO = n dimensional array containing initial condition vector x^o,

XADD = $(k \times n)$ matrix containing responses $x_i(t_j)$, XADD = $\{x_i(t_j)\}$, $i \in [1,k]$, $j \in [1,n]$,

RES = $(n \times k)$ matrix, transpose of XADD,

B = $(n \times n_d)$ matrix corresponding to the given matrix E,

M = integer corresponding to n_d.

13.4. Subroutines OUT2 and OUTN

Given cyclic (n × n) matrix A, n dimensional initial condition vector x^o, time increment Δt and number of intervals k, the subroutines calculate responses $x(t_i)$, $i \in [1,k]$, where the vector x(t) is defined by

$$\dot{x}(t) = Ax(t); \quad x(0) = x^o.$$

The calculation of x(t) is done by transforming given matrix A into the Jordan form with subroutine OUT2, and subsequent calculation of system modes, using subroutine OUTN.

Calling sequence for OUT

```
CALL OUT2(A,DT,N,K,XO,XADD,RES)
```

Definition of arguments

A = given (n × n) matrix,

DT = scalar corresponding to Δt,

N and K = integers corresponding to n and k,

XO = n dimensional array containing initial condition vector x^o,

XADD = (k × n) matrix containing responses $x_i(t_j)$, XADD = $\{x_j(t_j)\}$, $i \in [1,k]$, $j \in [1,n]$,

RES = (n × k) matrix, transpose of XADD.

Calling sequence for OUTN

```
CALL OUTN(Q,DT,RR,RI,N,K,XO,X)
```

Definition of arguments

Q = (n × n) matrix containing eigenvectors of the matrix A,

RR and RI = n dimensional arrays containing real and imaginary parts of the eigenvalues of the matrix A,

DT = scalar corresponding to Δt,

N and K = integers corresponding to n and k,

XO = n dimensional array, corresponding to initial condition vector x^o,

X = (n × k) matrix containing responses $x_i(t_j)$, $X = \{x_{ij}\}$, $x_{ij} = x_i(t_j)$.

Matrix X corresponds to the matrix RES in the calling sequence for the subroutine OUT2. Entries Q, RR and RI are calculated by the subroutine OUT2, similarly as in the subroutine JFORM.

Subroutine ORD

Given n dimensional arrays containing real and imaginary parts of n eigenvalues, $\lambda_i = \delta_i \pm j\omega_i$, the subroutine orders them in the descending order according to values of the real parts, i.e. $\delta_i \geq \delta_{i+1}$, $i \in [1,n-1]$.

Calling sequence

```
CALL ORD(RR,RI,N)
```

Description of arguments

RR and RI = n dimensional arrays containing respectively real and imaginary parts of n given eigenvalues,

N = integer corresponding to n.

After the execution of the subroutine the ordered real and imaginary parts appear in the same locations RR and RI, respectively. The reason for using this subroutine is that the IBM subroutines HSBG and ATEIG used for eigenvalue calculation, do not necessarily order the multiple eigenvalues consecutively.

14. LIBRARY AUXLIB

This section contains write-ups of some specialized subroutines called within the LINSYS. Detailed understanding of these subroutines is not necessary for LINSYS users.

15. REFERENCES

[1] Advanced Graphing II, User's Manual, Document No. 062-1530-00, Tektronix, Inc., October, 1973.

[2] System/360, Scientific Subroutine Package, Version III, Programmer's Manual, IBM Publication 360A-CM-03X, 1970.

[3] D. S. Schultz and J. L. Melsa, State Functions and Linear Control Systems, McGraw Hill Book Co., 1967.

[4] H. Kwakernaak and R. Sivan, Linear Optimal Control Systems, J. Wiley & Sons, Inc., 1972.

[5] D. L. Kleinman, "On an Iterative Technique for Riccati Equation Computations," IEEE Trans. on AC, Vol. AC-13, pp. 114-115, February, 1968.

[6] S. Bingulac and M. Stojic, "On an Iterative Solution of Time Invariant Riccati Equation," JACC Conf., Paper No. 3-C6, pp. 178-182, 1971.

[7] R. A. Smith, "Matrix Equation XA + BX = C," SIAM J. Appl. Math., Vol. 16, No. 1, 1968.

[8] M. L. J. Hautus, "Controllability and Observability Conditions of Linear Autonomous Systems," Proc. Kon. Ned. Akad. LXXII, No. 5, 1969, pp. 443-448.

[9] S. T. Chen, Introduction to Linear System Theory, Holt, Rinehart and Winston, New York, 1970.

[10] D. Djorovic and S. Bingulac, "Order Determination and Parameter Identification of Time Invariant State Variable Models," 5th IFAC Congress, Parix, 1972, Paper No. 38.5.

[11] F. R. Gantmacher, The Theory of Matrices, Chelsea, New York, 1960.

[12] J. L. Melsa and S. K. Jones, Computer Programs for Computational Assistance in the Study of Linear Control Theory, McGraw-Hill Book Co., 1970.

[13] S. Bingulac, "On the Calculation of Matrix Polynomials," IEEE Trans. on AC, Vol. AC-20, pp. 435-437, 1975.

[14] S. Bingulac, "On the Calculation of the Transfer Function Matrix," IEEE Trans. on AC, Vol. AC-20, pp. 134-136.

[15] H. W. Smith and E. J. Davison, "Design of Industrial Regulators - Integral Feedback and Feedforward Control," Proc. IEE, Vol. 119, No. 8, pp. 1210-1216, August, 1972.

16. EXAMPLES

A simple third order two-input, two-output system, Fig. 12, is chosen to illustrate the LINSYS use. The schematic of the physical model is given in Fig. 12a and a block diagram of its linearized model is given in Fig. 12b. The state variables are x_1, x_2, and x_3, the control inputs are u_1 and u_2, while w is the disturbance input. It is assumed that only x_2 and x_3 are available as outputs. Thus, the system matrices are

$$A = \begin{vmatrix} a_1 & 0 & 0 \\ 1 & a_2 & 0 \\ 0 & 1 & a_3 \end{vmatrix} ; \quad B = \begin{vmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{vmatrix} ; \quad C = \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}. \qquad (16.1)$$

For this system two design and one analysis problem are solved, each illustrating the use of LINSYS packages MRIC, MANAL and MPPL.

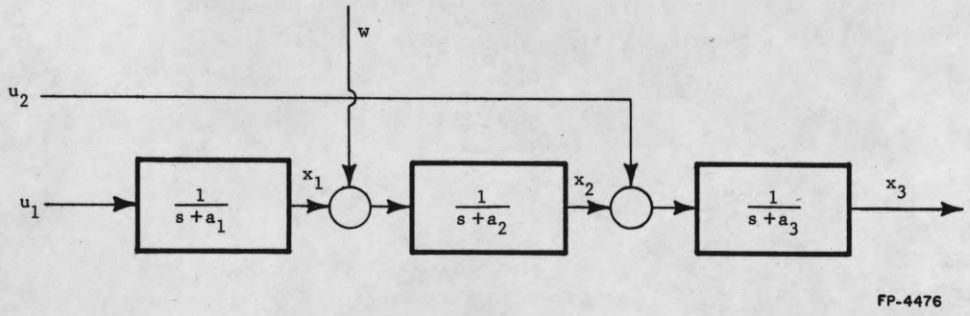
16.1. Problem #1. Optimal PI-Regulator Design - (MRIC)

The objective is to design an optimal PI-regulator which will guarantee zero-steady state errors for coordinates x_2 and x_3, whose desired values are x_{2s} and x_{3s}, i.e.

$$x_2(\infty) - x_{2s} = 0 \quad \text{and} \quad x_3(\infty) - x_{3s} = 0$$

Let $a_1 = -3$; $a_2 = -2$; $a_3 = -1$; $x_{2s} = 1$; $x_{3s} = 2$; $w = 2.5$. For the PI-regulator design [15] the following model is obtained

$$\hat{A} = \left|\begin{array}{c|c} A & 0 \\ \hline C & 0 \end{array}\right| = \begin{vmatrix} -3 & 0 & 0 & 0 & 0 \\ 1 & -2 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{vmatrix} \quad \hat{B} = \left|\begin{array}{c} B \\ \hline 0 \end{array}\right| = \begin{vmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{vmatrix}$$


w
u_2
u_1
$\frac{1}{s+a_1}$
x_1
$\frac{1}{s+a_2}$
x_2
$\frac{1}{s+a_3}$
x_3
FP-4476

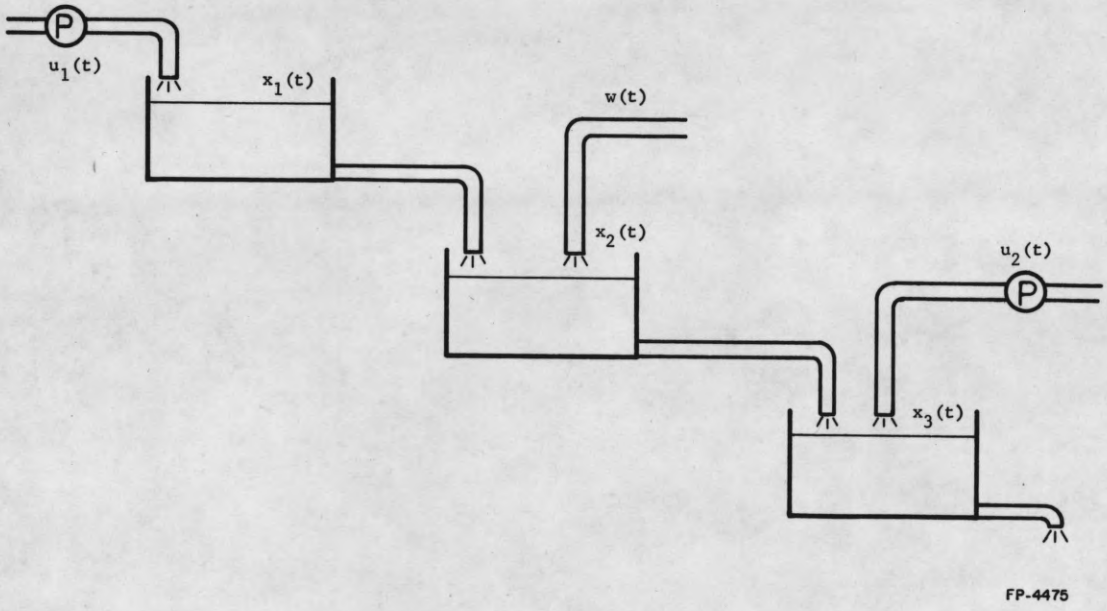

P
$u_1(t)$
$x_1(t)$
w(t)
$x_2(t)$
$u_2(t)$
P
$x_3(t)$
FP-4475

Figure 12a & b

$$z(t) = \begin{vmatrix} 0 \\ w \\ 0 \\ -x_{3s} \\ -x_{2s} \end{vmatrix} = \begin{vmatrix} 0 \\ 2.5 \\ 0 \\ -1 \\ -2 \end{vmatrix} 1(t).$$

The performance index to be minimized becomes

$$\int_0^\infty [x^T(t)\ C^{*T}\ C^* x(t) + u^T(t)\ Ru(t)]dt$$

where

$$C^* = \begin{vmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{vmatrix}; \quad \text{and} \quad R = \begin{vmatrix} 50 & 0 \\ 0 & 50 \end{vmatrix}.$$

The following pages contain the complete user-computer conversation during the solution of this problem.

```
.LOGIN 731,323
JOB 10 CSL 60189 TTY26
PASSWORD:
1758    10-JUN-75       TUE
.RU LINSYS
```

Starting the LINSYS execution

```
IN CASE OF USING DISK DATA FILE IT IS NECESSARY
TO ASSIGN DSK:2. IT MAY BE DONE BY TYPING:
<CTRL>C: <ASS DSK:2>,<CR> AND <CONT>,<CR>

 WANT SYSTEM BLOCK DIAGRAM - Y OR N
N
 TYPE <R>,<P>,<A>,<S> OR HELP
R
```

Reply 'R' - selection of MRIC package

```
 YOU ARE NOW ENTERING INTO THE RICCATI PACKAGE

          ENTER WITH N,M,K,IT,IE,IC
```

Integer input

```
5,2,2
     5     2     2     0     0     0
 SATISFIED WITH INPUT DATA - Y OR N
Y
AVAILABLE NAMES ARE <A  ><B  ><C  ><R  ><K  >
 TYPE <T>,<D>,<N> OR HELP
T
```

Matrix Input

Input from TTY

```
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
ALL
```

All matrices A, B, C, R and K entered via TTY

```
         A
-3
1,-2
,1,-1
,1
,,1
 -3.00  0.00  0.00  0.00  0.00
  1.00 -2.00  0.00  0.00  0.00
  0.00  1.00 -1.00  0.00  0.00
  0.00  1.00  0.00  0.00  0.00
  0.00  0.00  1.00  0.00  0.00
```

Subroutine INPDAT retypes entered values for each matrix

```
         B
1
,
,1
,
,
  1.00  0.00
  0.00  0.00
  0.00  1.00
  0.00  0.00
  0.00  0.00

         C
,,,1
,,,,1
  0.00  0.00  0.00  1.00  0.00
  0.00  0.00  0.00  0.00  1.00

         R
50
,50
 50.00  0.00
  0.00 50.00

         K
,
,
,
,
,
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00

 TYPE <T>,<D>,<N> OR HELP
N
```

```
  A       B       C       R       K
< 5, 5> < 5, 2> < 2, 5> < 2, 2> < 5, 5>
```

Names and dimensions of matrices typed by TYPDIM

```
AVAILABLE NAMES ARE <A  ><B  ><C  ><R  ><K  >
 TYPE <T>,<L>,<D>,<N> OR HELP
N
```

Matrix output - skipped

```
SYSTEM CONTROLLABLE
SYSTEM OBSERVABLE
 TYPE <I>,<E>,<C> OR HELP
C
          ERR
 0.41215E-04 0.65474E-08

 DO YOU WANT LOCATIONS OF EIG.VAL. Y OR N
N
AVAILABLE NAMES ARE <A  ><B  ><C  ><R  ><K  ><Q  >
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
K
          K
   1.30  3.91  0.86  7.01  0.90
   3.91 11.84  2.70 21.21  2.84
   0.86  2.70  6.57 -0.90  7.01
   7.01 21.21 -0.90 43.81 -0.89
   0.90  2.84  7.01 -0.89  7.95

 TYPE <T>,<L>,<D>,<N> OR HELP
D
TYPE NAMES OF MATRICES YOU WANT TO STORE ON DISK
OR TYPE EITHER <N> OR <ALL>
A,B,C,K
ENTER RECORD NAME FOR MATRIX <A  >  AINT
MATRIX <AINT > STORED IN REC # 72
ENTER RECORD NAME FOR MATRIX <B  >  BINT
MATRIX <BINT > STORED IN REC # 73
ENTER RECORD NAME FOR MATRIX <C  >  CIN
 RECORD WITH NAME <CIN  > ALREADY EXISTS
 TYPE <N>,<O>,<S> OR HELP
N
 ENTER NEW NAME  CINT
MATRIX <CINT > STORED IN REC # 74
ENTER RECORD NAME FOR MATRIX <K  >  KINT
MATRIX <KINT > STORED IN REC # 75
 TYPE <T>,<L>,<D>,<N> OR HELP
N\N\L
TYPE NAMES OF MATRICES YOU WANT TO PRINT ON LPT
OR TYPE EITHER <N> OR <ALL>
ALL
 TYPE <T>,<L>,<D>,<N> OR HELP
N
AVAILABLE NAMES ARE <RR ><RI ><FR ><AF >
 TYPE <T>,<L>,<D>,<N> OR HELP
L
TYPE NAMES OF MATRICES YOU WANT TO PRINT ON LPT
OR TYPE EITHER <N> OR <ALL>
ALL
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
FR
          FR
 -0.03 -0.08 -0.02 -0.14 -0.02
 -0.02 -0.05 -0.13  0.02 -0.14

 TYPE <T>,<L>,<D>,<N> OR HELP
D
TYPE NAMES OF MATRICES YOU WANT TO STORE ON DISK
OR TYPE EITHER <N> OR <ALL>
FR
ENTER RECORD NAME FOR MATRIX <FR >  FINT
MATRIX <FINT > STORED IN REC # 76
 TYPE <T>,<L>,<D>,<N> OR HELP
N
 WANT X(T),Y(T) & U(T) - Y OR N
N
 TYPE <I>,<E>,<R> OF HELP
R
```

Controllability and Observability tests

Block # 6 (Fig.3)
Values of the error quantity e_{rr}

Eigenvalues plotting skipped

Matrix output - block # 8

Riccati gain matrix K typed on TTY

A stored on disk record AINT

B stored on disk record BINT

C stored on disk record CINT

K stored on disk record KINT

All matrices (A, B, C, K and Q) printed on LPT

All available matrices (RR, RI, FR and AF) printed on LPT

FR typed on TTY

FR stored on disk record FINT

Plotting time domain responses skipped

Block # 10 - Return to LINSYS

16.2. Problem #2. Analysis of the Designed System (MANAL)

The objective is to analyze changes in pole locations and time-domain responses when instead of the proportional-integral gain

$$F = \left|\begin{array}{ccc:cc} -0.03 & -0.08 & -0.02 & -0.14 & -0.02 \\ -0.02 & -0.05 & -0.13 & 0.02 & -0.14 \end{array}\right|$$

obtained in the problem #1, only the integral part is used, i.e.

$$F^* = \left|\begin{array}{ccc:cc} 0.00 & 0.00 & 0.00 & -0.14 & -0.02 \\ 0.00 & 0.00 & 0.00 & 0.02 & -0.14 \end{array}\right| .$$

Since during the MRIC execution matrices A, B, and F have been stored on the disk data file, they will be read from DSK. The user will change the columns of F and thus obtain the matrix F*. The user-computer conversation during this analysis is given on the following pages

```
 TYPE <R>,<P>,<A>,<S> OR HELP
A
```

Reply 'A' - selection of MANAL package

```
YOU ARE NOW ENTERING INTO THE ANALYSIS PACKAGE

          ENTER N,M,K,IE
```

Integer input

```
5,2,5
     5     2     5     0
 SATISFIED WITH INPUT DATA - Y OR N
Y
AVAILABLE NAMES ARE <A  ><B  ><C  ><F  ><TM ><DT >
 TYPE <T>,<D>,<N> OR HELP
D
TYPE NAMES OF MATRICES YOU WANT TO READ FROM DISK
OR TYPE EITHER <N> OR <ALL>
A,B,C,F
 ENTER THE RECORD NAME FOR MATRIX <A  >  AINT
MATRIX <AINT > FOUND IN REC # 72
 ENTER THE RECORD NAME FOR MATRIX <B  >  BBINT
 RECORD WITH NAME <BBINT> DOES NOT EXIST
 ENTER THE RECORD NAME FOR MATRIX <B  >  BINT
MATRIX <BINT > FOUND IN REC # 73
 ENTER THE RECORD NAME FOR MATRIX <C  >  SKIP.
 ENTER THE RECORD NAME FOR MATRIX <F  >  FINT
MATRIX <FINT > FOUND IN REC # 76
 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
C,F
 IN <C  > YOU WANT TO CHANGE E,R,C,D,M OR N
H
E , R , C , D , M & N  STAND FOR
ELEMENT, ROW, COLUMN, DIAGONAL (MAIN), MATRIX (WHOLE) & NONE
 IN <C  > YOU WANT TO CHANGE E,R,C,D,M OR N
D
          C
,,,,1,1
TOO MANY ELEMENTS IN THE ROW - TRY AGAIN
,,,1,1
  0.00   0.00   0.00   1.00   1.00

 IN <F  > YOU WANT TO CHANGE E,R,C,D,M OR N
C
          TYPE INDICES OF EL.ROW OR COL
1
 1  1
          F
D
 I BEG YOUR PARDON                    D
0
  0.00   0.00

 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
F
 IN <F  > YOU WANT TO CHANGE E,R,C,D,M OR N
C
          TYPE INDICES OF EL.ROW OR COL
2
 2  1
          F
,
  0.00   0.00

 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
F
 IN <F  > YOU WANT TO CHANGE E,R,C,D,M OR N
C
          TYPE INDICES OF EL.ROW OR COL
3
 3  1
          F
0,
  0.00   0.00

 TYPE <T>,<D>,<N> OR HELP
N
```

Matrix input

Reading from DSK

Matrix A - record name AINT

Typing error

Matrix B - record name BINT

Reading of matrix C skipped

Matrix F - record name FINT

Input from TTY

Reply 'H' - additional information

Only main diagonal of C is entered

Typing error

Values accepted

Reduction of F to F* (see problem # 2)

First column

Typing error

Second column

Third column

Exit from INPDAT subroutine

```
  A       B       C       F       TM      DT
< 5, 5> < 5, 2> < 5, 5> < 2, 5> < 5, 5> < 1, 1>

AVAILABLE NAMES ARE <A  ><B  ><C  ><F  ><TM ><DT >
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
A,B,C,F
          A
 -3.00  0.00  0.00  0.00  0.00
  1.00 -2.00  0.00  0.00  0.00
  0.00  1.00 -1.00  0.00  0.00
  0.00  1.00  0.00  0.00  0.00
  0.00  0.00  1.00  0.00  0.00

          B
  1.00  0.00
  0.00  0.00
  0.00  1.00
  0.00  0.00
  0.00  0.00

          C
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  1.00  0.00
  0.00  0.00  0.00  0.00  1.00

          F
  0.00  0.00  0.00 -0.14 -0.02
  0.00  0.00  0.00  0.02 -0.14

 TYPE <T>,<L>,<D>,<N> OR HELP
N
 TYPE <I>,<E>,<C> OR HELP
C


 DO YOU WANT LOCATIONS OF EIG.VAL. Y OR N
N
AVAILABLE NAMES ARE <AF ><RR ><RI >
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
ALL
          AF
 -3.00  0.00  0.00 -0.14 -0.02
  1.00 -2.00  0.00  0.00  0.00
  0.00  1.00 -1.00  0.02 -0.14
  0.00  1.00  0.00  0.00  0.00
  0.00  0.00  1.00  0.00  0.00

          RR
 -3.04 -1.94 -0.82 -0.17 -0.02

          RI
  0.00  0.00  0.00  0.00  0.00

 TYPE <T>,<L>,<D>,<N> OR HELP
N
 WANT CONT./OBS. TEST ?
N\N\Y
SYSTEM CONTROLLABLE
SYSTEM OBSERVABLE
 TYPE <D>,<C>,<N> OR HELP
N
 WANT X(T),Y(T) & U(T) - Y OR N
N
 TYPE <J>,<A>,<N> OR HELP
N
 TYPE <I>,<E>,<P>,<R> OR HELP
R
 WANT SYSTEM BLOCK DIAGRAM - Y OR N
N
 TYPE <R>,<P>,<A>,<S> OR HELP
S
END OF EXECUTION
CPU TIME: 3.07  ELAPSED TIME: 6:2.53
EXIT
.
```

Matrix names and dimensions typed by TYPDIM

Matrix Output

Matrices A, B, C and F typed on TTY

Block # 4 -(Fig.5) option Continue

Eigenvalues plotting skipped

Block # 6
All available matrices typed on TTY

Controllability and Observability tests

Return to LINSYS
Reply 'R'

Reply 'S' - Stop

16.3. Problem #3. Pole Placement and Observer Design (MPPL)

The objective is for the system (16.1) to find a feedback gain matrix F which places the closed loop poles at

-1.5, -2.5 and -3.5.

Since the complete state is not measurable, the state observer is to be designed. For observer poles the following values are assumed

-6, -8, -10.

The user-computer conversation during the solution of this problem is given on the following pages.

```
 TYPE <R>,<P>,<A>,<S> OR HELP
P

 YOU ARE NOW ENTERING INTO THE POLE PLACEMENT/ /OBSERVER PACKAGE

          ENTER WITH N,M,K,IE
3,2,2
     3     2     2     0
 SATISFIED WITH INPUT DATA - Y OR N
Y
AVAILABLE NAMES ARE <A  ><B  ><RP ><IP >
 TYPE <T>,<D>,<N> OR HELP
ALL
?A? - TRY AGAIN
 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
ALL
          A
-3
1,-2
,1,-1
 -3.00  0.00  0.00
  1.00 -2.00  0.00
  0.00  1.00 -1.00

          B
1
,
,1
  1.00  0.00
  0.00  0.00
  0.00  1.00

          RP
-1.5,-2.5,-3.5
 -1.50 -2.50 -3.50

          IP
,
  0.00  0.00  0.00

 TYPE <T>,<D>,<N> OR HELP
N

  A        B        RP       IP
< 3, 3> < 3, 2> < 1, 3> < 1, 3>

AVAILABLE NAMES ARE <A  ><B  ><RP ><IP >
 TYPE <T>,<L>,<D>,<N> OR HELP
N
 TYPE <I>,<E>,<C> OR HELP
C
CONTROLLABILITY/OBSERVABILITY INDICES  2  1
ORDERED CONTROLLABILITY/OBSERVABILITY INDICES  1  2

AVAILABLE NAMES ARE <A  ><B  ><RP ><IP ><F  ><AF >
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
F,AF
          F
 -1.00 -0.75  0.00
  0.00 -1.00 -0.50

          AF
 -4.00 -0.75  0.00
  1.00 -2.00  0.00
  0.00  0.00 -1.50

 TYPE <T>,<L>,<D>,<N> OR HELP
L
TYPE NAMES OF MATRICES YOU WANT TO PRINT ON LPT
OR TYPE EITHER <N> OR <ALL>
ALL
 TYPE <T>,<L>,<D>,<N> OR HELP
N
```

Reply 'P' - selection of MPPL package

Integer input

Typing error

Matrix input - All matrices (A, B, RP and IP) entered via TTY

Subroutine INPDAT retypes entered values for each matrix

Matrix names and dimensions typed by TYPDIM

Block # 4 (Fig.4) Option Continue

Block # 5 - Pole placement controllability indices typed

Matrix output - block # 6

F and AF typed on TTY

All available matrices printed on LPT

Exit from OUTDAT subroutine Reply 'N'

```
 DO YOU WANT STATE OBSERVER - Y OR N
Y
AVAILABLE NAMES ARE <C  ><RO ><IO >
 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
C,RO
         C
,1
,,1
  0.00  1.00  0.00
  0.00  0.00  1.00

         RO
-6,-8,-10
 -6.00 -8.00-10.00

 TYPE <T>,<D>,<N> OR HELP
N

  C       RO      IO
< 2, 3> < 1, 3> < 1, 3>

AVAILABLE NAMES ARE <C  ><RO ><IO >
 TYPE <T>,<L>,<D>,<N> OR HELP
D
TYPE NAMES OF MATRICES YOU WANT TO STORE ON DISK
OR TYPE EITHER <N> OR <ALL>
C,RO
ENTER RECORD NAME FOR MATRIX <C  >  COP1
MATRIX <COP1 > STORED IN REC # 81
ENTER RECORD NAME FOR MATRIX <RO >  ROP1
MATRIX <ROP1 > STORED IN REC # 82
 TYPE <T>,<L>,<D>,<N> OR HELP
N
 TYPE <I>,<E>,<C> OR HELP
C
CONTROLLABILITY/OBSERVABILITY INDICES  2  1
ORDERED CONTROLLABILITY/OBSERVABILITY INDICES  1  2

AVAILABLE NAMES ARE <C  ><RO ><IO ><K  ><AOF>
 TYPE <T>,<L>,<D>,<N> OR HELP
L
TYPE NAMES OF MATRICES YOU WANT TO PRINT ON LPT
OR TYPE EITHER <N> OR <ALL>
ALL
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
K,AOF
         K
-35.00  0.00
-13.00  0.00
 -1.00 -5.00

         AOF
 -3.00-35.00  0.00
  1.00-15.00  0.00
  0.00  0.00 -6.00

 TYPE <T>,<L>,<D>,<N> OR HELP
N
```

Block # 7 - Reply 'Y' selects Observer design

Matrix Input

Matrices C and RO entered via TTY
IO asumed zero value

Matrix names and dimensions typed by TYPDIM

Matrix Output - block # 9

C stored on disk record COP1

RO stored on disk record ROP1

Block # 11 - Observer design
Observability indices typed

Matrix Output - block # 12

All available matrices printed on LPT

K and AOF typed on TTY

```
AVAILABLE NAMES ARE <AT ><RT ><IT >
 TYPE <T>,<L>,<D>,<N> OR HELP
T
TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY
OR TYPE EITHER <N> OR <ALL>
ALL
          AT
 -4.00 -0.75  0.00  1.00  0.75  0.00
  1.00 -2.00  0.00  0.00  0.00  0.00
  0.00  0.00 -1.50  0.00  1.00  0.50
  0.00  0.00  0.00 -3.00-35.00  0.00
  0.00  0.00  0.00  1.00-15.00  0.00
  0.00  0.00  0.00  0.00  0.00 -6.00

          RT
 -3.50 -2.50 -1.50-10.00 -8.00 -6.00

          IT
  0.00  0.00  0.00  0.00  0.00  0.00

 TYPE <T>,<L>,<D>,<N> OR HELP
N

 DO YOU WANT LOCATIONS OF EIG.VAL. Y OR N
N
 WANT X(T),Y(T) & U(T) - Y OR N
N
 TYPE <I>,<P>,<O>,<R> OR HELP
P
AVAILABLE NAMES ARE <A  ><B  ><RP ><IP >
 TYPE <T>,<D>,<N> OR HELP
N

  A        B        RP       IP
< 3, 3> < 3, 2> < 1, 3> < 1, 3>

AVAILABLE NAMES ARE <A  ><B  ><RP ><IP >
 TYPE <T>,<L>,<D>,<N> OR HELP
D
TYPE NAMES OF MATRICES YOU WANT TO STORE ON DISK
OR TYPE EITHER <N> OR <ALL>
A,B,RP
ENTER RECORD NAME FOR MATRIX <A  >  APP1
MATRIX <APP1 > STORED IN REC # 78
ENTER RECORD NAME FOR MATRIX <B  >  BPP1
MATRIX <BPP1 > STORED IN REC # 79
ENTER RECORD NAME FOR MATRIX <RP >  RPP1
MATRIX <RPP1 > STORED IN REC # 80
 TYPE <T>,<L>,<D>,<N> OR HELP
N
 TYPE <I>,<E>,<C> OR HELP
C
CONTROLLABILITY/OBSERVABILITY INDICES  2  1
ORDERED CONTROLLABILITY/OBSERVABILITY INDICES  1
```

Matrices AT, RT and IT typed on TTY

Plotting of eigenvalues and time domain responses skipped - replies 'N'

Block # 14 - In order to store matrices on DSK, control is transfered to block # 2 - reply 'P'

A stored on disk record APP1

B stored on disk record BPP1

RP stored on disk record RPP1

```
.RU EDITD

 TYPE <N>,<M>,<D>,<S> OR HELP
H
TO TYPE/PRINT NAMES
              MATRICES;
              DELETE/INSERT/CHANGE RECORDS; TO
              STOP OR FOR
              HELP
 TYPE <N>,<M>,<D>,<S> OR HELP
N
TYPE <L> OR <T>
T
 NAME =  ARIC1  NUMBER =     1
 NAME =  BRIC1  NUMBER =     2
 NAME =  QRIC1  NUMBER =     3
 NAME =  RRIC1  NUMBER =     4
 NAME =  ARIC2  NUMBER =     5
 NAME =  BRIC2  NUMBER =     6
 NAME =  QRIC2  NUMBER =     7
 NAME =  RRIC2  NUMBER =     8
 NAME =  KRIC2  NUMBER =     9
 NAME =  BRAN   NUMBER =    10
 NAME =  EINT   NUMBER =    65
 NAME =  ASK    NUMBER =    66
 NAME =  AAA    NUMBER =    67
 NAME =  BBB    NUMBER =    68
 NAME =  CCC    NUMBER =    69
 NAME =  KKK    NUMBER =    70
 TYPE <N>,<M>,<D>,<S> OR HELP
M
TYPE <L> OR <T>
L
 TYPE <N>,<M>,<D>,<S> OR HELP
D
 TYPE -1 FOR A NAME TO EXIT
 ENTER MATRIX NAME  CDSYS
 TYPE,DELETE,CHANGE,NONE, - T,D,C,OR N
D
 ENTER MATRIX NAME  BDSYS
 TYPE,DELETE,CHANGE,NONE, - T,D,C,OR N
DTDSY
 ENTER MATRIX NAME  DTDSY
 TYPE,DELETE,CHANGE,NONE, - T,D,C,OR N
D
 ENTER MATRIX NAME  -1
 DID YOU DO ANY DELETIONS? - Y OR N
Y
TYPE: DEL NAMES.DAT,MATRIC.DAT

      REN NAMES.DAT=TNAM.DAT

      REN MATRIC.DAT=TMAT.DAT

STOP

END OF EXECUTION
CPU TIME: 29.48 ELAPSED TIME: 5:58.08
EXIT

.DEL NAMES.DAT,MATRIC.DAT
FILES DELETED:
NAMES.DAT
MATRIC.DAT
74 BLOCKS FREED

.REN NAMES.DAT=TNAM.DAT
FILES RENAMED:
TNAM.DAT

.REN MATRIC.DAT=TMAT.DAT
FILES RENAMED:
TMAT.DAT

.PRINT *.LPT
TOTAL OF 26 BLOCKS IN 1 FILE IN LPT REQUEST

.K/F
JOB 3, USER [731,323]  LOGGED OFF TTY26    1107  14-JUN-75
SAVED ALL FILES (1290 BLOCKS)
CPU TIME: 1 MIN, 3.90 SEC        CONNECT TIME: 30 MIN    KCS: 811
```

Starting EDITD execution

Additional information typed
Reply 'H'

Selection of the Block # 2
Reply 'N'

Record names typed on TTY
Reply 'T'

(only first 10 and last 6
names shown)

Selection of the Block # 3
Reply 'M'

Matrices and record names printed
on LPT - Reply 'L'

Selection of the Block # 4
Reply 'D'

Records with names
CDSYS, BDSYS and DTSYS
deleted

Message typed by EDITD

End of EDITD execution

Monitor commands - transfering
edited data from temporary files
into files accessible by LINSYS
packages

Monitor command - printing
results on LPT

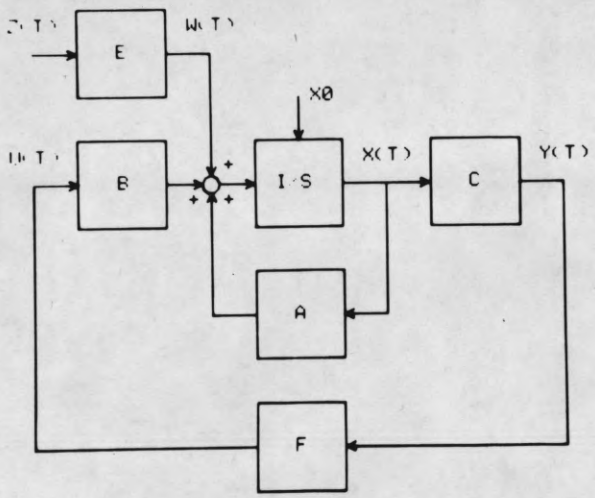
"Logging off"

Example of user-computer conversations within plotting subroutines BLDG, EGVP and PLXU

```
.C
 EX F10@CLINS1
LINK     LOADING
[LNKXCT LINSYS EXECUTION]

IN CASE OF USING DISK DATA FILE IT IS NECESSARY
TO ASSIGN DSK:2  IT MAY BE DONE BY TYPING:
<CTRL>C) <ASS DSK:2>,<CR> AND <CONT>,<CR>

 WANT SYSTEM BLOCK DIAGRAM - Y OR N
Y

 SYSTEM BLOCK DIAGRAM
```

Starting the LINSYS execution

System block diagram plotted by BLDG

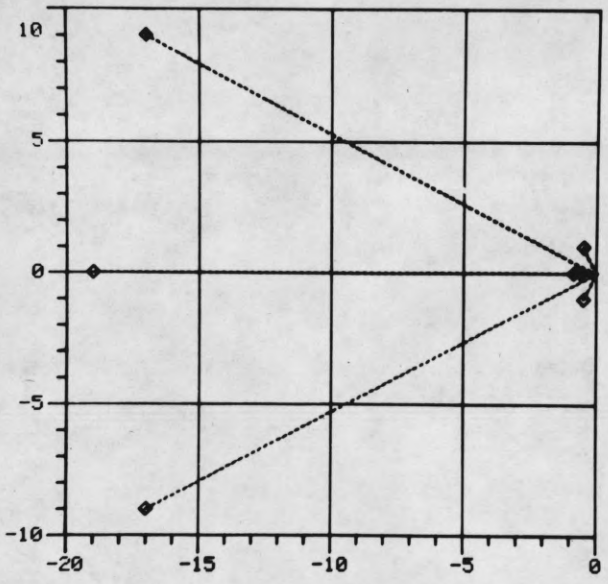


```
 TYPE <R>,<P>,<A>,<S> OR HELP
A

YOU ARE NOW ENTERING INTO THE ANALYSIS PACKAGE

          ENTER N,M,K,IC
7
     7     0     0     0
 SATISFIED WITH INPUT DATA - Y OR N
Y
AVAILABLE NAMES ARE <A  ><B  ><C  ><F  ><TM ><DT >
 TYPE <T>,<D>,<N> OR HELP
D
TYPE NAMES OF MATRICES YOU WANT TO READ FROM DISK
OR TYPE EITHER <N> OR <ALL>
A
 ENTER THE RECORD NAME FOR MATRIX <A  >  ADEM
MATRIX <ADEM > FOUND IN REC # 72
 TYPE <T>,<D>,<N> OR HELP
N

  A       B       C       F       TM      DT
< 7, 7> < 7, 1> < 1, 7> < 1, 1> < 7, 7> < 1, 1>

AVAILABLE NAMES ARE <A  ><B  ><C  ><F  ><TM ><DT >
 TYPE <T>,<L>,<D>,<N> OR HELP
N
 TYPE <I>,<E>,<C> OR HELP
C
 DO YOU WANT LOCATIONS OF EIG.VAL. Y OR N
Y
```

Selection of MANAL package
Reply 'A'

Matrix A read from the
disk data file
Record name ADEM

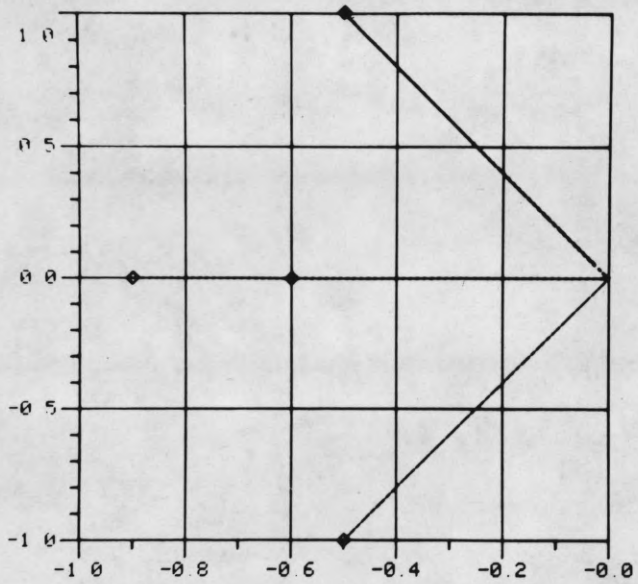
Execution of the EGVP
subroutine - Reply 'Y'



Part of the s-plane containing
all eigenvalue locations

```
 DO YOU WANT NEW PLOT
Y
WHICH EIG VAL  YOU WANT TO ELIMINATE
TYPE LIM  VALUE
          R-R
-0 50000E+00-0.50000E+00-0.60000E+00-0 90000E+00-0.19000E+02
-0 17000E+02-0.17000E+02

          RR LIM
-2
-0 20000E+01

          RED.R-R
-0 50000E+00-0.50000E+00-0.60000E+00-0.90000E+00

          RED.R-I
 0 10000E+01-0 10000E+01 0.00000E+00 0.00000E+00
```

Only dominant eigenvalues with δ_i > RR.LIM = -2 plotted



Exit from EGVP
Reply 'N'

```
 DO YOU WANT NEW PLOT
N
AVAILABLE NAMES ARE <AF ><RR ><RI >
 TYPE <T>,<L>,<D>,<N> OR HELP
N
 WANT CONT./OBS. TEST ?  .
N
 TYPE <D>,<C>,<N> OR HELP
H
TO OBTAIN DISCRETE TRANSF.FUN.MATRIX' (STATE TRANSIT.MAT);
          CONTIN.  TRANSF.FUN.MATRIX; OR
          NONE
 TYPE <D>,<C>,<N> OR HELP
N
 WANT X(T),Y(T) & U(T) - Y OR N
Y
```

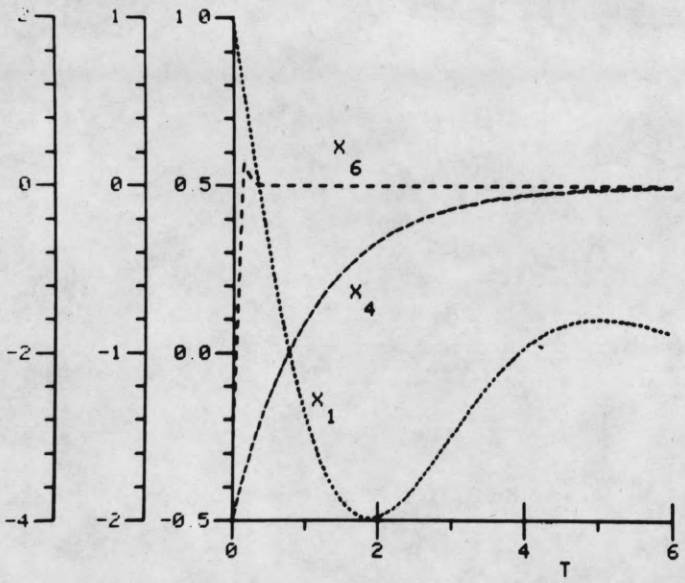
Continuation of the MANAL package

```
 WANT X(T),Y(T) & U(T) - Y OR N
Y
AVAILABLE NAMES ARE <ND ><E  ><X0 ><TT >
 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
X0,TT
          X0
1,-1,2,-2,3,-3,4
  1 00 -1.00  2.00 -2.00  3.00 -3.00  4.00

          TT
6
  6 00

 TYPE <T>,<D>,<N> OR HELP
N

  ND        E         X0        TT
< 1, 1> < 7, 0> < 1, 7> < 1, 1>

AVAILABLE NAMES ARE <ND ><E  ><X0 ><TT >
 TYPE <T>,<L>,<D>,<N> OR HELP
N
# OF POINTS =   42
AVAILABLE NAMES ARE <ND ><E  ><X0 ><TT ><RES>
 TYPE <T>,<L>,<D>,<N> OR HELP
N
          TYPE INDICES YOU WANT TO PLOT
1,4,6
  1 00  4 00  6 00  0 00  0 00  0.00  0 00
```

Execution of PLXU subroutine - Reply 'Y'

Input of initial conditions (XO) and total time (TT)

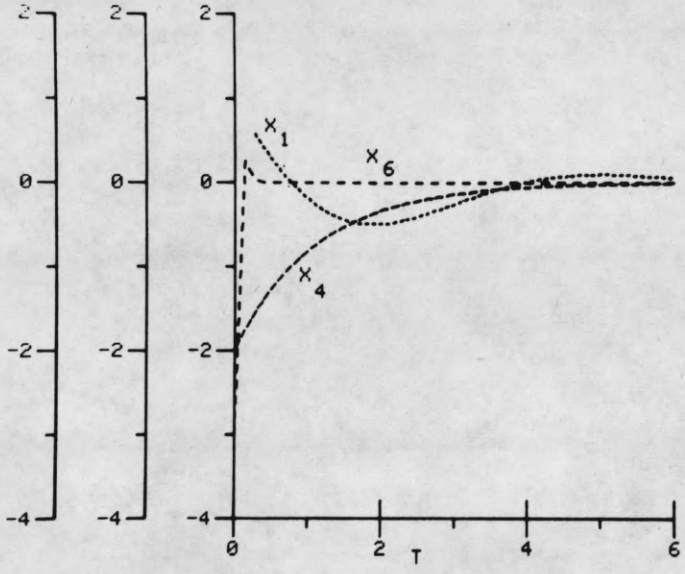
Indices of x(t) are entered



Obtained plot

```
Y
 DO YOU WANT NEW PLOT WITH COMMON SCALE - Y OR N
          YMIN,YMAX
-4,2
 -4.00  2.00
```

Same plot with common scale
$y_{min} = -4$; $y_{max} = 2$



```
 DO YOU WANT NEW PLOT WITH COMMON SCALE - Y OR N
N
 DO YOU WANT Y(T) - Y OR N
N
 DO YOU WANT U(T) - Y OR N
N
 TYPE <E>,<I>,<N> OR HELP
E
AVAILABLE NAMES ARE <ND ><E  ><X0 ><TT >
 TYPE <T>,<D>,<N> OR HELP
T
TYPE MATRIX NAMES YOU WANT TO CHANGE OR ENTER VIA TTY
OR TYPE EITHER <N> OR <ALL>
TT
         TT
 4
  0.40

 TYPE <T>,<D>,<N> OR HELP
N

  ND       E        X0       TT
< 1, 1> < 7, 0> < 1, 7> < 1, 1>

AVAILABLE NAMES ARE <ND ><E  ><X0 ><TT >
 TYPE <T>,<L>,<D>,<N> OR HELP
N
# OF POINTS =   42
         TYPE INDICES YOU WANT TO PLOT
1,4,6,7
  1 00  4.00  6.00  7.00  0.00  0.00  0.00
```

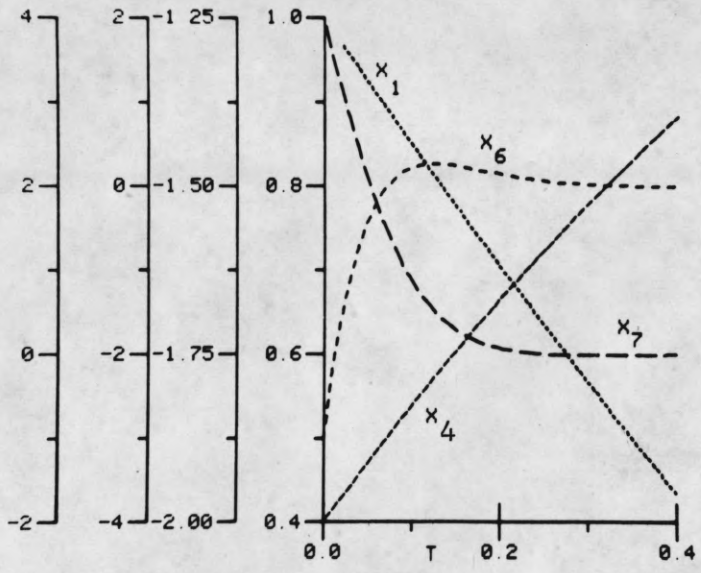
Continuation of PLXU subroutine

Return to the beginning of PLXU
Reply 'E'

New total time (TT) is entered

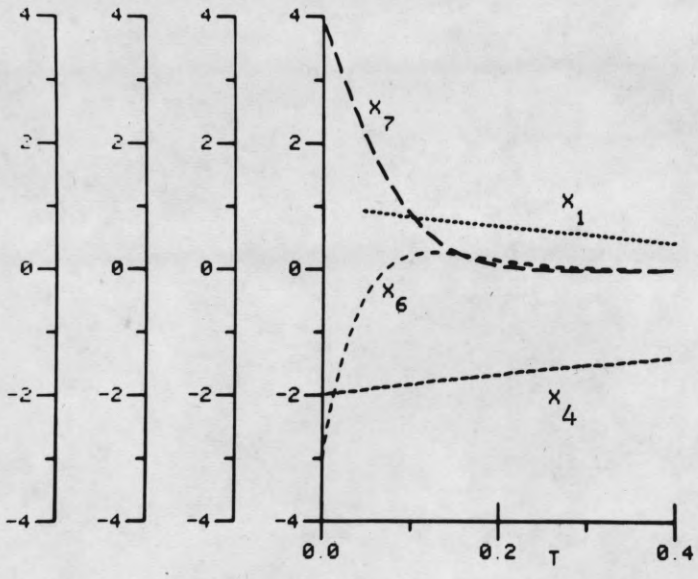
New indices of x(t) are entered

Obtained plot



```
 DO YOU WANT NEW PLOT WITH COMMON SCALE - Y OR N
Y
          YMIN,YMAX
-4,4
 -4.00  4.00
```

Same plot with common scale
$y_{min} = -4$; $y_{max} = 4$



```
 DO YOU WANT NEW PLOT WITH COMMON SCALE - Y OR N
N
 DO YOU WANT Y(T) - Y OR N
N
 DO YOU WANT U(T) - Y OR N
N
 TYPE <E>,<I>,<N> OR HELP
N
 TYPE <J>,<A>,<N> OR HELP
N
 TYPE <I>,<E>,<P>,<R> OR HELP
R
 WANT SYSTEM BLOCK DIAGRAM - Y OR N
N
 TYPE <R>,<P>,<A>,<S> OR HELP
S
STOP
END OF EXECUTION
CPU TIME:  22.28 ELAPSED TIME:  22:6.67
EXIT
.K/F
JOB 16, USER [731,323]  LOGGED OFF TTY14    1436  16-JUN-75
SAVED ALL FILES (1210 BLOCKS)
CPU TIME:  55.24 SEC     CONNECT TIME:  42 MIN    KCS:  2071
```

Continuation of PLXU subroutine

Exit from PLXU - Reply 'N'

Exit from Manal - Reply 'R'

Stop of the LINSYS execution - Reply 'S'

"Logging off"

17. LINSYS EXECUTION AND USAGE

LINSYS software may be executable by the monitor command

```
.RU LINSYS[731,323]
```

LINSYS.SAV is a name of the "SAVE" file on the disk area [731,323] containing all necessary routines for LINSYS execution. If user wants to introduce modifications into some of the packages or subroutines he may execute the modified LINSYS either by the monitor command:

```
.EX@CLINS[731,323]
```

or by

```
.EX/F10@CLINS1[731,323]
```

CLINS.CMD and CLINS1.CMD are names of "COMMAND" files containing names of all files necessary for LINSYS execution. The file CLINS.CMD uses F40 FORTRAN compiler, while CLINS1.CMD uses F10. The file LINSYS.SAV is created using the F10 compiler. Files CLINS.CMD and CLINS1.CMD are as follows:

```
.
TYPE CLINS.CMD
[731,323]LINSYS.F4,LPPL/SEA,LBLDG/SEA
[731,323]LRIC/SEA,LOUT/SEA,LGEN/SEA,SYS:AG240/SEA
[731,323]LGEN1/SEA,AUXLIB/SEA
[731,323]LDISK/SEA,LIOM/SEA,SYS:SSP40/SEA
.
.TYPE CLINS1.CMD
[731,323]LINSYS.F4,LPPL/SEA,LBLDG/SEA
[731,323]LRIC/SEA,LOUT/SEA,LGEN/SEA,OLD:AG210/SEA
[731,323]GEN1/SEA,AUSLIB/SEA
[731,323]LDISK/SEA,LIOM/SEA,AUXLIB/SEA,SYS:SSP10/SEA
```

The need for alternate compilers is that in some particular cases, due to yet unknown reasons, either one or the other compiler does not give satisfactory results.

In order to minimize core memory requirement, some of the subroutines use COMMON statements. In an independent use of some of these subroutines it might become necessary to rearrange the COMMON statements or to substitute them by individual DIMENSION statements. The package EDITD is executable either by the monitor command

```
.RU EDITD[731,323]
```

or by

```
EX@CEDITD[731,323]
```

The "command" file CEDITD.CMD is of the form:

```
TYPE CEDITD.CMD
[731,323]EDITD.F4,LIOM/SEA,LDISK/SEA
```

All files* whose names appear in the listed .CMD files exist in the disk area [731,323] and are accessible for modification and transfer to other disk areas.

The names of all files and the subroutines contained within each file are listed in the following table.

---

* With the exception of system "SYS:" files containing the SSP Package and the AG-II software.

```
FORTRAN: LINSYS
MAIN.
MANAL
MPPL
MRIC
FORTRAN: LPPL
PPL
RED
FAD
EVCHE
FORTRAN: LBLDG
BLDG
AR
LN
SQ
CR
FORTRAN: LRIC
RIC
LYAP
TEST
TRC
STT
STABK
JACC
FORTRAN: LOUT
PLXU
SEL
CALYU
EGVP
MRLOC
RLOC
PLOTN
RESP
OUT2
OUTN
ORD
CURSE
FORTRAN: LGEN
RANKR
JFORM
COIN2
COMF
EGT
TRANS
FRT
CHEQ
RANK2
HERMIT
DET
JOIN
DJOIN
FORTRAN: LGEN1
MTF
FRT
DREAL
OUT
FORTRAN: AUXLIB
CHT1
FFA
FFB
FFC
FFD
AV
CHEQ
CHT
MATPL
FORTRAN: LDISK
INPDAT
INPDSK
HLP
OUTDAT
DSKRD
ADISK
OUTLPT
OUTTTY
OUTDSK
DSKWR
IFALL
RCHAR
TYPDIM
FORTRAN: LIOM
INPTTY
CHANGE
IOMT
IM
INTT
LOC
ININ
IFF
IFFM
PRINT
CALDIM
FORTRAN: AUXLIB
CHT1
FFA
FFB
FFC
FFD
AV
CHEQ
CHT
MATPL
```

18. PROGRAM LISTINGS

Index of Listings

```
F40I    V27(360)        12-JUN-75       14:32   PAGE 1


C               CONVERSATIONAL SOFTWARE   L I N S Y S
C               FOR ANALYSIS AND DESIGN OF LINEAR SYSTEMS
C
C               EXECUTABLE BY:    .RU LINSYS=731,323=
C               OR                .EX@CLINS=731,323=
C
      COMMON UCC(1600)
      COMMON/MP/ UCM(1600)
101   FORMAT('  TO RUN RICCATI PACKAGE;'/
     1'         POLE PLACEMENT/OBSERVER PACKAGE;'/
     2'         ANALYSIS PACKAGE; OR TO'/
     3'         STOP'/)
102   FORMAT('  YOU ARE NOW ENTERING INTO THE RICCATI PACKAGE'/)
103   FORMAT('  YOU ARE NOW ENTERING INTO THE POLE PLACEMENT'
     1' /OBSERVER PACKAGE'/)
104   FORMAT(/' YOU ARE NOW ENTERING INTO THE ANALYSIS PACKAGE'/)
105   FORMAT(1X,A1)
      IBL=' '
      TYPE 105,IBL
      CALL HLP(1)
      TYPE 105,IBL
6     CALL IFF($7,$8,35,' WANT SYSTEM BLOCK DIAGRAM = Y OR N')
7     CALL BLOG
8     CALL SCLA(UCC,0.,40,40,0)
      CALL SCLA(UCM,0.,40,40,0)
      GO TO 9
5     TYPE 101
9     CALL IFFM($1,$2,$3,$4,$5,'R','P','A','S','H',29,
     1' TYPE <R>,<P>,<A>,<S> OR HELP')
1     TYPE 105,IBL
      TYPE 102
      TYPE 105,IBL
      CALL MRIC
      GO TO 6
2     TYPE 105,IBL
      TYPE 103
      TYPE 105,IBL
      CALL MPPL
      GO TO 6
3     TYPE 105,IBL
      TYPE 104
      TYPE 105,IBL
      CALL MANAL
      GO TO 6
4     STOP
      END
```

```
C                RICCATI PACKAGE

      SUBROUTINE MRIC
      COMMON/MP/ A(100),B(100),Q(100),R(100),AK(100),AF(100)
     1,AKM(100),RM(100),RR(10),RI(10),F(100),C(100),UCM(580)
      DIMENSION NVI(12),NVVI(8),NVV(12)
      DIMENSION ITF(6),ITP(4),NV(12)
      COMMON L(10),L1(10),M1(10),UC(1570)
      DATA ITF/'A','B','C','R','K','Q'/,ITP/'RR','RI','FR','AF'/
      DATA NVI/1,1,1,2,3,1,2,2,1,1,1,1/,NVVI/3,1,3,1,2,1,1,1/
30    CALL ININ(6,26,' ENTER WITH N,M,K,IT,IE,IC',N,M,K,IT,IE,IC)
100   FORMAT(10I3)
      IRR=1
      IPL=0
      ICT=0
      IF(M.EQ.0) M=1
      IF(K.EQ.0) K=N
      IF(IT.EQ.0)IT=10
      IF(IE.GE.0) IE=-6
      IF(IC.GE.0) IC=-2
      EPS=10.**IE
      EP1=10.**IC
      IF(N.EQ.0) GO TO 30
      CALL CALDIM(NVI,NV,12,N,M,K,1)
      CALL CALDIM(NVVI,NVV,8,N,M,1,1)
      IR=1
3     CONTINUE
      CALL INPDAT(5,NV,ITF,A,B,C,RM,AKM,AD,IRR)
      CALL TYPDIM(5,NV,ITF)
      CALL OUTDAT(5,NV,ITF,A,B,C,RM,AKM,AD)
      CALL MCPY(RM,R,M,M,0)
      IF(IR.EQ.1) GO TO 33
11    CALL IFF($32,$33,43,' WANT TO USE OBTAINED K AS IN,GUESS= Y OR N')
33    CALL MCPY(AKM,AK,N,N,0)
32    CONTINUE
      CALL EGVP(A,N,RR,RI,1)
      CALL RANKR(A,B,RR,RI,N,M,EP1,IRC,'C')
      CALL RANKR(A,C,RR,RI,N,K,EP1,IRO,'O')
      IF(IRC.NE.0) TYPE 111
      IF(IRO.NE.0) TYPE 112
111   FORMAT(' SYSTEM CONTROLLABLE')
112   FORMAT(' SYSTEM OBSERVABLE')
      GO TO 5
6     CALL HLP(2)
5     CALL IFFM($30,$3,$31,$6,$6,'I','E','C','H','H',25,
     1' TYPE <I>,<E>,<C> OR HELP')
31    CALL GTPRD(C,C,Q,K,N,N)
      CALL RIC(A,B,Q,R,AK,EPS,N,M,IT,IJ,AF)
      IF(IJ.EQ.0) GO TO 3
      CALL GTPRD(B,AK,UC,N,M,N)
      CALL GMPRD(R,UC,F,M,M,N)
      CALL SMPY(F,-1.,F,M,N,0)
      IR=IR+1
      CALL EGVP(AF,N,RR,RI,IPL)
      CALL OUTDAT(6,NV,ITF,A,B,C,R,AK,Q)
      CALL OUTDAT(4,NVV,ITP,RR,RI,F,AF,AD,AD)
      IF(IPL.EQ.0) CALL PLXU(AF,B,R,AK,N,M,1,'RC')
      GO TO 1
2     CALL HLP(3)
1     CALL IFFM($30,$3,$4,$2,$2,'I','E','R','H','H',25,
     1' TYPE <I>,<E>,<R> OF HELP')
4     RETURN
      END
```

```
C                POLE PLACEMENT/OBSERVER PACKAGE

      SUBROUTINE MPPL
      COMMON UC(1100),A21(100),AT(100),P1(100),P2(100),D(100)
      COMMON/MP/ A(100),B(100),RR(10),RI(10),AD(100)
     1,FR(100),AF(100),NV(10),NZ(10),L1(10),M1(10),LV(10)
     2,AFP(100),RRO(10),RIO(10),RR1(10),RI1(10),A12(100),C(790)
      DIMENSION ITF(6),ND(20),ITF1(5),NDD(20),NVO(10),NVP(12)
     1,NVJI(6),ITJ(3),NVJ(6)
      DATA ITF1/'C','RO','IO','K','AOF'/,NVO/3,1,4,1,4,1,1,3,1,1/
     1,ITJ/'AT','RT','IT'/
      DATA ITF/'A','B','RP','IP','F','AF'/
     1,NVP/1,1,1,2,4,1,4,1,2,1,1,1/,NVJI/1,1,4,1,4,1/
2     CALL ININ(4,20,' ENTER WITH N,M,K,IE',N,M,K,IE,L,L)
      IR=1
      IRR=1
      IF(N.LE.1) GO TO 2
      IF(K.EQ.0) K=1
      IF(M.EQ.0) M=1
      IF(IE.EQ.0) IE=-2
      EPS=10.**IE
100   FORMAT(10I3)
      CALL CALDIM(NVP,ND,12,N,M,K,1)
      CALL CALDIM(NVO,NDD,10,N,M,K,1)
24    CALL INPDAT(4,ND,ITF,A,B,RR1,RI1,A1,A2,IR)
      CALL TYPDIM(4,ND,ITF)
      CALL OUTDAT(4,ND,ITF,A,B,RR1,RI1,A1,A2)
      GO TO 9
8     CALL HLP(2)
9     CALL IFFM($2,$24,$14,$8,$8,'I','E','C','H','H',25,
     1' TYPE <I>,<E>,<C> OR HELP')
14    CALL MCPY(RR1,RR,1,N,0)
      CALL MCPY(RI1,RI,1,N,0)
      CALL PPL(A,B,RR,RI,AD,N,M,EPS,FR,P1,AFP,ICO,'C')
      IF(ICO.EQ.1) GO TO 24
      JPL='PP'
      N1=N
      CALL MCPY(AFP,AT,N,N,0)
      CALL MCPY(P1,A12,N,N,0)
      CALL OUTDAT(6,ND,ITF,A,B,RR,RI,FR,AFP)
      CALL IFF($4,$5,36,' DO YOU WANT STATE OBSERVER = Y OR N')
4     CALL INPDAT(3,NDD,ITF1,C,RRO,RIO,A1,A2,A3,IRR)
      CALL TYPDIM(3,NDD,ITF1)
      CALL OUTDAT(3,NDD,ITF1,C,RRO,RIO,A1,A2,A3)
      GO TO 10
11    CALL HLP(2)
10    CALL IFFM($2,$4,$3,$11,$11,'I','E','C','H','H',25,
     1' TYPE <I>,<E>,<C> OR HELP')
3     CALL MCPY(RRO,RR,1,N,0)
      CALL MCPY(RIO,RI,1,N,0)
      CALL PPL(A,C,RR,RI,AD,N,K,EPS,D,P1,AF,ICO,'O')
      IF(ICO.EQ.1) GO TO 4
      CALL OUTDAT(5,NDD,ITF1,C,RR,RI,D,AF,A1)
      CALL SCLA(A21,0.,N,N,0)
      CALL JOIN(AFP,A12,A21,AF,N,N,N,N,N1,N1,AT)
      CALL EGVP(AT,N1,RR,RI,1)
      CALL CALDIM(NVJI,NVJ,6,N1,1,1,1)
      CALL OUTDAT(3,NVJ,ITJ,AT,RR,RI,A1,A2,A3)
      JPL='PO'
5     CALL EGVP(AT,N1,RR,RI,0)
      CALL PLXU(AT,FR,P1,P2,N1,M,1,JPL)
      GO TO 1
6     TYPE 101
1     CALL IFFM($2,$24,$4,$7,$6,'I','P','O','R','H',29,
     1' TYPE <I>,<P>,<O>,<R> OR HELP')
101   FORMAT(' TO CHANGE INTEGERS; DATA FOR'/
     1'            POLE PLACEMENT; DATA FOR'/
     2'            OBSERVER DESIGN; OR TO'/
     3'            RETURN TO LINSYS')
7     RETURN
      END
```

```
C
C                ANALYSIS PACKAGE

      SUBROUTINE MANAL
      COMMON UCC(1600)
      COMMON/MP/ A(100),B(100),C(100),F(100),V(100),RR(10),RI(10)
     1,UC(100),UCM(270)
      DIMENSION NAMEI(6),NDC(12),ND(12),NDC2(12),ND2(12),NAME2(3)
     2,ITC(4),ITT(5)
      COMMON/MP/ FF(100),G(100),CHE(10),TFM(100),AT(100)
     3,BT(100),CT(100),TM(100)
      DATA NDC/1,1,1,2,3,1,2,3,1,1,4,4/
      DATA NDC2/1,1,4,1,4,1,1,3,1,1,1,2/
      DATA NAMEI/'A','B','C','F','TM','DT'/,NAME2/'AF','RR','RI'/
     1,ITC/'CHE','TFM','F','G'/,ITT/'AT','BT','CT','FT','TM'/
1     CALL ININ(4,15,' ENTER N,M,K,IE',N,M,K,IE,L,L)
      IR=1
      IF(IE.GE.0) IE=-3
      EPS=10.**IE
      IF(M.LE.0) M=1
      IF(K.LE.0) K=1
      CALL CALDIM(NDC,ND,12,N,M,K,1)
      CALL CALDIM(NDC2,ND2,12,N,M,K*M,1)
2     CALL INPDAT(6,ND,NAMEI,A,B,C,F,TM,DT,IR)
      CALL TYPDIM(6,ND,NAMEI)
      CALL OUTDAT(6,ND,NAMEI,A,B,C,F,TM,DT)
      CALL MCPY(A,V,N,N,0)
      CALL MCPY(B,BT,N,M,0)
      CALL MCPY(C,CT,K,N,0)
      GO TO 4
5     CALL HLP(2)
4     CALL IFFM($1,$2,$3,$5,$5,'I','E','C','H','H',25,
     1' TYPE <I>,<E>,<C> OR HELP')
3     CALL GMPRD(B,F,UC,N,M,K)
      CALL GMPRD(UC,C,V,N,K,N)
      CALL GMADD(A,V,V,N,N)
21    CALL EGVP(V,N,RR,RI,0)
      CALL OUTDAT(3,ND2,NAME2,V,RR,RI,A1,A2,A3)
      CALL MCPY(V,FF,N,N,0)
      CALL MCPY(B,G,N,M,0)
      CALL IFF($11,$34,23,' WANT CONT./OBS. TEST ?')
11    CONTINUE
      CALL RANKR(V,B,RR,RI,N,M,EPS,IRC,'C')
      CALL RANKR(V,C,RR,RI,N,K,EPS,IRO,'O')
      IF(IRC.NE.0) TYPE 111
      IF(IRO.NE.0) TYPE 112
111   FORMAT(' SYSTEM CONTROLLABLE')
112   FORMAT(' SYSTEM OBSERVABLE')
      GO TO 34
9     CALL HLP(5)
34    CALL IFFM($6,$7,$8,$9,$9,'O','C','N','H','H',25,
     1' TYPE <O>,<C>,<N> OR HELP')
6     CALL DREAL(A,B,N,M,DT,FF,G,'O')
7     CALL MTF(FF,G,C,N,M,K,0,CHE,WD,TFM)
      CALL OUTDAT(4,ND2(5),ITC,CHE,TFM,FF,G,AD,AD)
8     IF(IFL.EQ.0) CALL PLXU(V,B,F,C,N,M,K,'AN')
      GO TO 13
12    CALL HLP(4)
13    CALL IFFM($19,$10,$16,$12,$12,'J','A','N','H','H',25,
     1' TYPE <J>,<A>,<N> OR HELP')
19    CALL JFORM(V,BT,CT,N,M,K,UC,FF)
      GO TO 14
10    CALL TRANS(V,BT,CT,TM,UC,N,M,K,1)
      CALL MCPY(TM,UC,N,N,0)
14    CALL OUTDAT(5,ND,ITT,V,BT,CT,F,UC,AD)
      GO TO 16
15    CALL HLP(6)
16    CALL IFFM($1,$2,$8,$17,$15,'I','E','P','R','H',29,
     1' TYPE <I>,<E>,<P>,<R> OR HELP')
17    RETURN
      END
```

```
C       EDITD.F4  EXTENDED MATRIX DISK FILE HANDLER
C       EXECUTABLE WITH MONITOR COMMAND  .EX/F10@CEDITD


      INTEGER RECNUM
      DIMENSION REC(2),A(100),RECOR(103),AZ(103),REC0(2)
      DIMENSION AA(2),NV(2)
      EQUIVALENCE(NAM,REC(1))
      EQUIVALENCE(RECNUM,REC(2))
      EQUIVALENCE(RECOR(4),A)
      GO TO 25
26    TYPE 109
109   FORMAT(' TO TYPE/PRINT NAMES',/
     1'                MATRICES;',/
     2'                DELETE/INSERT/CHANGE RECORDS; TO',/
     3'                STOP OR FOR',/
     4'                HELP')
25    CALL IFFM($21,$22,$20,$11,$26,'N','M','D','S','H',29,
     1' TYPE <N>,<M>,<D>,<S> OR HELP')
21    CALL PNAMES
      GO TO 25
22    CALL PRIDAT
      GO TO 25
20    CALL DEFINE FILE (1,3,NXT,'NAMES',0,0)
      CALL DEFINE FILE (2,103,NZ,'MATRIC',0,0)
      TYPE 108
108   FORMAT('  TYPE -1 FOR A NAME TO EXIT')
      RECNUM = 0
      NMAT = 1
1     IR=1
      TYPE 101
101   FORMAT('  ENTER MATRIX NAME  ',$)
      ACCEPT 102,NAME
102   FORMAT(A5)
      IF(NAME.EQ.'-1')GO TO 9
10    READ(1#IR,END=90)REC
      IF(NAME.EQ.NAM)GO TO 3
      IF(NAM.EQ.0) GO TO 90
      IR=NXT
      GO TO 10
90    TYPE 103,NAME
103   FORMAT('  IS ',A5,' NEW?')
      CALL IFF($2,$1,14,' ANSWER Y OR N')
2     CALL IOMT(AA,1,2,-1,24,' ENTER MATRIX DIMENSIONS')
      N=AA(1)
      M=AA(2)
      IF(N.LE.0) GO TO 1
      IA=N*M
      NV(1)=N
      NV(2)=M
      JR=1
      CALL INPTTY(A,A1,A2,A3,A4,A5,'MAT',NV,NMAT,JR)
      IF(NAM.EQ.0) IR = IR-1
      READ(1#(IR-1),END=91)REC
91    RECNUM=RECNUM+1
      WRITE(1#IR)NAME,RECNUM
      RECOR(1)=IA
      RECOR(2)=N
      RECOR(3)=M
      WRITE(2#RECNUM)RECOR
      GO TO 1
3     CALL IFFM($5,$6,$7,$8,$8,'T','D','C','N','N',42,' TYPE,
     1DELETE,CHANGE,NONE, - T,D,C,OR N')
5     READ(2#RECNUM)RECOR
      N=RECOR(2)
      M=RECOR(3)
      TYPE 106,NAME,N,M
106   FORMAT('  NAME = ',A5,' DIMENSIONS = ',I3,2X,I3)
      CALL IOMT(A,N,M,1,1,'A')
      GO TO 3
6     IR=IR+1
      READ(1#IR,END=92)REC
      IF(NAM.EQ.0) GO TO 92
      READ (2#RECNUM)RECOR
      RECNUM=RECNUM-1
      WRITE(1#(IR-1))REC
      WRITE(2#RECNUM)RECOR
      GO TO 6
92    WRITE(1#(IR-1))REC0
      WRITE(2#RECNUM)AZ
      GO TO 1
7     READ(2#RECNUM)RECOR
      NV(1)=RECOR(2)
      NV(2)=RECOR(3)
      JR=2
      CALL INPTTY(A,A1,A2,A3,A4,A5,'MAT',NV,NMAT,JR)
      WRITE(2#RECNUM)RECOR
      GO TO 3
8     GO TO 1
9     CALL IFF($14,$11,35,' DID YOU DO ANY DELETIONS? - Y OR N')
14    CALL DEFINE FILE(3,3,NX,'TNAM',0,0)
      CALL DEFINE FILE(4,103,NY,'TMAT',0,0)
      II=1
13    READ(1#II,END=12)REC
      IF(NAM.EQ.0) GO TO 12
      READ(2#RECNUM)RECOR
      WRITE(3#II)REC
      WRITE(4#RECNUM)RECOR
      II=NXT
      GO TO 13
12    TYPE 110
110   FORMAT(' TYPE: DEL NAMES.DAT,MATRIC.DAT'/
     1'       REN NAMES.DAT=TNAM.DAT'/
     2'       REN MATRIC.DAT=TMAT.DAT'/)
11    STOP
      END
```

```
      SUBROUTINE PRIDAT
C       PRINTS THE CONTENTS OF DATA FILES FOR MATRICES

      DIMENSION A(100),RECOR(103),REC(2)
      EQUIVALENCE (NAME,REC(1))
      EQUIVALENCE (NUM,REC(2))
      EQUIVALENCE(RECOR(4),A)
      CALL DEFINE FILE(1,3,NXT,'NAMES',0,0)
      CALL DEFINE FILE (2,103,NZ,'MATRIC',0,0)
2     TYPE 101
101   FORMAT(' TYPE <L> OR <T>')
      ACCEPT 102,IT
      IF(IT.NE.'T'.AND.IT.NE.'L') GO TO 2
102   FORMAT(A1)
      K=1
      DO 1 I=1,200
      READ(1#K,END=90)REC
      K=NXT
      READ(2#NUM)RECOR
      N=RECOR(2)
      M=RECOR(3)
      IF(IT.EQ.'L') PRINT 100,NAME,NUM,N,M
      IF(IT.EQ.'T') TYPE  100,NAME,NUM,N,M
100   FORMAT('  NAME = ',A5,' RECORD# = ',I3,' DIMENSIONS = 'I3,2X
     12X,I3)
      IF(IT.EQ.'L') CALL PRINT(A,N,M,2,5,NAME)
      IF(IT.EQ.'T') CALL IOMT(A,N,M,3,5,NAME)
1     CONTINUE
90    RETURN
      END
```

```
      SUBROUTINE PNAMES
      INTEGER REC(2)
      EQUIVALENCE(NAME,REC(1))
      EQUIVALENCE(NUM,REC(2))
      CALL DEFINE FILE(1,3,NXT,'NAMES',0,0)
101   FORMAT(A1)
2     TYPE 102
102   FORMAT(' TYPE <L> OR <T>')
      ACCEPT 101,IPR
      IF(IPR.NE.'T'.AND.IPR.NE.'L') GO TO 2
      DO 12 I=1,100
      READ(1#I,END=90)REC
      IF(IPR.EQ.1HT) TYPE  100,NAME,NUM
      IF(IPR.EQ.1HL) PRINT 100,NAME,NUM
100   FORMAT('  NAME =  ',A5,'  NUMBER =  ',I3)
12    CONTINUE
90    RETURN
      END
```

```
      SUBROUTINE PLXU(A,B,R,AK,N,M,IK,IND)
      DIMENSION A(1),B(1),R(1),AK(1)
      DIMENSION X0(10),X00(10),ED(100),EDD(100)
     1,ITF(5),ND(10),NDC(10)
      COMMON L1(10),M1(10),E(100),DK(100),UCC(200),UC(600)
      DATA ITF/'ND','E','X0','TT','RES'/
     1,NDC/4,4,1,2,4,1,4,4,1,3/
100   FORMAT(' # OF POINTS =',I5)
C           IND = 'RC'/'PP'/'PO'/ AND 'AN'
      INEW=0
      IR=1
      CALL IFF($32,$7,31,' WANT X(T),Y(T) & U(T) = Y OR N')
32    CALL CALDIM(NDC,ND,8,N,5,8,1)
5     CALL INPDAT(4,ND,ITF,AL,ED,X0,TT,A1,A2,IR)
      L=AL
      CALL CALDIM(NDC,ND,10,N,L,8,1)
      CALL TYPDIM(4,ND,ITF)
      CALL OUTDAT(4,ND,ITF,AL,ED,X0,TT,A1,A2)
      KM=100
      IUC=301
      NL=N
      IF(L.GT.N) NL=L
      K=300/NL
      IF(K.GT.KM) K=KM
      TYPE 100,K
      DT=TT/(K-1.)
17    CALL MCPY(X0,X00,1,N,0)
      ND(10)=K
      INEW=INEW+1
      CALL MCPY(ED,EDD,N,L,0)
      CALL RESP(A,DT,N,K,X00,UC,UC(IUC),EDD,L)
      IF(INEW.EQ.1) CALL OUTDAT(5,ND,ITF,AL,ED,X0,TT,UC(IUC),A2)
      E(1)=0.
      DO 8 I=2,K
8     E(I)=E(I-1)+DT
      N1=N
      CALL SEL($42,N,UC,K,N1,IUC)
      CALL PLOTN(N1,K,E,UC(IUC))
42    IR=1
      IF(IND.EQ.'AN') IR=2
      DO 10 II=1,IR
      IO=II
      N1=M
      IF(II.EQ.1.AND.IND.EQ.'AN') N1=IK
      CALL CALYU($10,B,R,AK,DK,UC,UC(IUC),N,M,IK,K,IO,IND)
      CALL SEL($10,N1,UC,K,N1,IUC)
      CALL PLOTN(N1,K,E,UC(IUC))
10    CONTINUE
      GO TO 40
6     CALL HLP(7)
40    CALL IFFM($32,$17,$7,$6,$6,'E','I','N','H','H',25,
     1' TYPE <E>,<I>,<N> OR HELP')
7     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EGVP(A,N,RR,RI,IND)
      DIMENSION A(1),RR(1),RI(1)
     1,C(100)
      DIMENSION L1(20),M1(10),B(100)
     1,D(100)
C       FOR IND.NE.0, NO PLOT
      IHSBG=1
      IRIC=1
      CALL SCLA(C,0.,N,N,0)
      IF(IHSBG.EQ.0) GO TO 3
      CALL MCPY(A,C,N,N,0)
      CALL HSBG(N,C,N)
      CALL ATEIG(N,C,RR,RI,L1,N)
      GO TO 4
      CALL CHEQ(A,N,B,C,D)
3     B(N+1)=1.
      CALL POLRT(B,C,N,RR,RI,IE)
      IF(IE.NE.0) TYPE 100,IE
4     IF(IRIC.EQ.1) GO TO 5
C       CALL IOMT(B,1,N,1,5,'CH.EQ')
      CALL IOMT(RR,1,N,1,2,'RR')
      CALL IOMT(RI,1,N,1,2,'RI')
100   FORMAT(10I3)
5     IF(IND.NE.0) RETURN
      CALL MRLOC(N,RR,RI,B,C)
      RETURN
      END
```

```
      SUBROUTINE BLDG
      DIMENSION ISQ(14)
     1,    ILN(68)
     1,    ICH(32),NC(16),NAM(32),K(16)
     1,    ICR(2)
     1,    IAR(20),KR(10),IND(10)
      DATA ISQ/200,350,400,350,600,350,400,200,400,50,
     1200,500,200,50/
      DATA ILN/150,400,200,400,300,400,340,400,
     1   360,400,400,400,500,400,600,400,
     2   700,400,750,400,750,400,750,100,
     3   750,100,500,100,400,100,150,100,
     4   150,100,150,400,550,400,550,250,
     5   550,250,500,250,400,250,350,250,
     6   350,250,350,390,150,550,200,550,
     7   300,550,350,550,350,550,350,410,
     8   450,500,450,450/
      DATA ICH/450,250,250,400,650,400,250,550,
     1   450,100,470,500,430,400,530,430,
     2   730,430,130,430,130,580,180,380,
     3   370,380,330,380,370,420,330,580/
      DATA NC/1,1,1,1,1,2,3,4,4,4,4,1,1,1,1,4/
      DATA NAM/65,66,67,69,70,88,48,73,47,83,88,40,84,41,
     1   89,40,84,41,85,40,84,41,90,40,84,41,32,43,87,40,84,41/
      DATA K/1,2,3,4,5,6,8,11,15,19,23,27,28,28,28,29/
      DATA ICR/350,400/
      DATA IAR/200,400,400,400,600,400,500,250,
     1   500,100,200,550,450,450,350,400,
     2   350,400,350,400/,
     3    KR/1,1,1,1,1,1,1,10,10,10/,
     4    IND/'R','R','R','L','L','R','D','U','R','D'/
      CALL INITT(30)
      CALL BINITT
      NAR=10
      NCR=1
      NCH=16
      NL=17
      NSQ=6
      I3=5
      NR=10
      DO 7 I=1,30
      TYPE 101,IB
      IB=' '
7     CONTINUE
      IBL=' '
101   FORMAT('+',A1)
      TYPE 102
102   FORMAT(' SYSTEM BLOCK DIAGRAM')
      DO 1 I=1,NSQ
      J=2*I-1
1     CALL SQ(ISQ(J),ISQ(J+1),100,100)
      DO 2 I=1,NL
      J=4*I-3

2     CALL LN(ILN(J),ILN(J+1),ILN(J+2),ILN(J+3))
      DO 3 I=1,NCH
      J=2*I-1
      KK=K(I)
3     CALL NOTATE(ICH(J)-I3,ICH(J+1)-I3,NC(I),NAM(KK))
      DO 4 I=1,NCR
      J=2*I-1
4     CALL CR(ICR(J),ICR(J+1),NR)
      DO 5 I=1,NAR
      J=2*I-1
5     CALL AR(IAR(J),IAR(J+1),KR(I),IND(I))
      CALL TINPUT(LL)
      RETURN
      END
```

```
      SUBROUTINE AR(J1,J2,N,IND)
      I1=J1
      I2=J2
      IF(IND.EQ.'R') I1=I1-N
      IF(IND.EQ.'L') I1=I1+N
      IF(IND.EQ.'U') I2=I2-N
      IF(IND.EQ.'D') I2=I2+N
      ID2=10
      ID1=ID2/2
      IF(IND.EQ.'U'.OR.IND.EQ.'D') GO TO 2
      IF(IND.EQ.'R') ID2=-ID2
      CALL LN(I1,I2,I1+ID2,I2+ID1)
      CALL LN(I1,I2,I1+ID2,I2-ID1)
      RETURN
2     IF(IND.EQ.'U') ID2=-ID2
      CALL LN(I1,I2,I1+ID1,I2+ID2)
      CALL LN(I1,I2,I1-ID1,I2+ID2)
      RETURN
      END


      SUBROUTINE CR(I1,I2,N)
      CALL MOVABS(I1,I2+N)
      DO 2 I=1,N
      P=3.1415*2.
      X=(I-1.)/(N-1.)*P
      J1=N*SIN(X)+I1
      J2=N*COS(X)+I2
      CALL DRWABS(J1,J2)
2     CONTINUE
      RETURN
      END


SUBROUTINE SQ(X1,Y1,DX,DY)
INTEGER X1,Y1,DX,DY
CALL MOVABS(X1,Y1)
CALL DRWABS(X1+DX,Y1)
CALL DRWABS(X1+DX,Y1+DY)
CALL DRWABS(X1,Y1+DY)
CALL DRWABS(X1,Y1)
RETURN
END


SUBROUTINE LN(X1,Y1,X2,Y2)
INTEGER X1,X2,Y1,Y2
CALL MOVABS(X1,Y1)
CALL DRWABS(X2,Y2)
RETURN
END
```

```
      SUBROUTINE INPOAT(NM,NV,ITF,A,B,C,D,E,F,IR)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),NV(1),ITF(1)
      COMMON XXZ(100)
      IO='<'
      IC='>'
      TYPE 102,(IO,ITF(I),IC,I=1,NM)
      GO TO 5
6     TYPE 101
101   FORMAT(' FOR CHANGING OR READING MATRICES FROM TTY, DSK,OR NONE')
102   FORMAT(' AVAILABLE NAMES ARE ',6(A1,A3,A1))
5     CALL IFFM($1,$2,$6,$4,$4,'T','D','H','N','N',25,
     1 ' TYPE <T>,<D>,<N> OR HELP')
1     CALL INPTTY(A,B,C,D,E,F,ITF,NV,NM,IR)
      IR=IR+1
      GO TO 5
2     CALL INPDSK(A,B,C,D,E,F,NV,NM,ITF)
      IR=IR+1
      GO TO 5
4     RETURN
      END


      SUBROUTINE INPDSK(A,B,C,D,E,F,NV,NM,ITF)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),NV(1),ITF(1),IT(10)
     1,F1(1),F2(1)
6     CONTINUE
      TYPE 102
102   FORMAT(' TYPE NAMES OF MATRICES YOU WANT TO READ FROM DISK',
     1/' OR TYPE EITHER <N> OR <ALL>')
      CALL RCHAR(IT,1,NM)
      N=NM
      CALL IFALL($10,$7,NM,IT,ITF)
7     DO 1 I=1,N
      JJ=N-I+1
      IF(IT(JJ).EQ.'N') RETURN
1     IF(IT(JJ).NE.' ')GO TO 2
2     CONTINUE
      N=JJ
10    CONTINUE
      DO 3 I=1,N
      DO 4 J=1,NM
      JS=J
4     IF(IT(I).EQ.ITF(J)) GO TO 5
      TYPE 101,IT(I)
101   FORMAT(' MATRIX WITH NAME <',A3,'>  DOES NOT EXIST - TRY AGAIN')
      GO TO 6
5     CONTINUE
      KS=JS*2-1
      GO TO (21,22,23,24,25,26),JS
26    CALL DSKRD(1,F,F1,F2,NV(KS),ITF(JS))
      GO TO 3
25    CALL DSKRD(1,E,F1,F2,NV(KS),ITF(JS))
      GO TO 3
24    CALL DSKRD(1,D,F1,F2,NV(KS),ITF(JS))
      GO TO 3
23    CALL DSKRD(1,C,F1,F2,NV(KS),ITF(JS))
      GO TO 3
22    CALL DSKRD(1,B,F1,F2,NV(KS),ITF(JS))
      GO TO 3
21    CALL DSKRD(1,A,F1,F2,NV(KS),ITF(JS))
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ININ(NI,NT,IT,I,J,K,L,M,N)
      DIMENSION A(6),IA(6),IT(10)
10    CONTINUE
      CALL IOMT(A,1,NI,0,NT,IT)
      DO 8 IJ=1,NI
8     IA(IJ)=A(IJ)
      GO TO (1,2,3,4,5,6),NI
6     N=A(6)
5     M=A(5)
4     L=A(4)
3     K=A(3)
2     J=A(2)
1     I=A(1)
      TYPE 101,(IA(IJ),IJ=1,NI)
101   FORMAT(1X,10I6)
      CALL IFF($7,$10,35,' SATISFIED WITH INPUT DATA - Y OR N')
7     RETURN
      END
```

```
      SUBROUTINE INPTTY(A,B,C,D,E,F,ITF,NV,NMAT,ITER)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),ITF(1),NV(1)
     1,IT(10)
      DIMENSION IOP(5),AA(100)
C       COMMON AA(100)
      DATA IOP/'E','R','C','D','M'/
      IA='M'
      GO TO 8
C       IF(ITER.NE.1) GO TO 8
13    DO 9 I=1,NMAT
9     IT(I)=ITF(I)
      N=NMAT
      ITER=1
      IA='M'
      GO TO 10
8     CONTINUE
6     CONTINUE
      TYPE 102
102   FORMAT(' TYPE MATRIx NAMES YOU WANT TO CHANGE OR ENTER VIA TTY',
     1/' OR TYPE EITHER <N> OR <ALL>')
104   FORMAT(10I3)
100   FORMAT(10A3)
      CALL RCHAR(IT,1,NMAT)
      N=NMAT
      DO 1 I=1,N
      JJ=N-I+1
      IF(IT(I).EQ.'ALL') GO TO 13
      IF(IT(JJ).EQ.'N') RETURN
1     IF(IT(JJ).NE.' ')GO TO 2
2     CONTINUE
      N=JJ
      IF(JJ.LE.NMAT) GO TO 10
      TYPE 106,NMAT
106   FORMAT(' ONLY',I3,'  NAMES ALOWED')
      GO TO 6
10    CONTINUE
      DO 3 I=1,N
      DO 4 J=1,NMAT
      JS=J
4     IF(IT(I).EQ.ITF(J)) GO TO 5
      TYPE 101,IT(I)
101   FORMAT(' MATRIX WITH NAME <',A3,'>  DOES NOT EXIST - TRY AGAIN')
      GO TO 6
5     CONTINUE
      I2=JS*2
      IF(NV(I2-1).NE.1.OR.NV(I2).NE.1) GO TO 14
      ITER=1
      IA='M'
14    CONTINUE
      IF(ITER.NE.1)     TYPE 103,IT(I)
103   FORMAT('  IN <',A3,'> YOU WANT TO CHANGE E,R,C,D,M OR N')
      IR=1
      IC=1
      J1=2*JS-1
      J2=J1+1
109   FORMAT(A1)
      IF(ITER.NE.1)     ACCEPT 109,IA
      IF(IA.NE.'M') GO TO 15
      TYPE 108
108   FORMAT(' E , R , C , D , M & N  STAND FOR'/
     1' ELEMENT, ROW, COLUMN, DIAGONAL (MAIN), MATRIX (WHOLE) & NONE'/)
      GO TO 5
15    IF(IA.EQ.'N') GO TO 3
      DO 18 JJ=1,5
18    IF(IA.EQ.IOP(JJ)) GO TO 19
      TYPE 107,IA
107   FORMAT(' OPTION ',A1,' DOES NOT YET EXIST - TRY AGAIN')
      GO TO 5
19    CONTINUE
      IF(IA.EQ.'M') IR=NV(J1)
      IF(IA.EQ.'C'.OR.IA.EQ.'D') IC=NV(J1)
      IF(IA.EQ.'R'.OR.IA.EQ.'M') IC=NV(J2)
      IF(IA.EQ.'D') IC=NV(J1)
C     N=MAX(N,M)
      IF(IA.EQ.'M'.OR.IA.EQ.'D') GO TO 7
11    CALL IOMT(AA,1,2,0,30,' TYPE INDICES OF EL,ROW OR COL')
      I1=AA(1)
      I2=AA(2)
      IF(I2.EQ.0) I2=1
      TYPE 104,I1,I2
      I1M=NV(J1)
      IF(IA.EQ.'C') I1M=NV(J2)
      IF(I1.LE.I1M.AND.I2.LE.NV(J2)) GO TO 12
105   FORMAT(' INDICES OUT OF BOUNDS')
      TYPE 105
      GO TO 11
12    CONTINUE
      IF(I1.LE.0) GO TO 11
7     CONTINUE
      CALL IOMT(AA,IR,IC,-1,1,IT(I))
      GO TO (21,22,23,24,25,26),JS
21    CALL CHANGE(A,AA,IA,I1,I2,NV(1),NV(2))
      GO TO 3
22    CALL CHANGE(B,AA,IA,I1,I2,NV(3),NV(4))
      GO TO 3
23    CALL CHANGE(C,AA,IA,I1,I2,NV(5),NV(6))
      GO TO 3
24    CALL CHANGE(D,AA,IA,I1,I2,NV(7),NV(8))
      GO TO 3
25    CALL CHANGE(E,AA,IA,I1,I2,NV(9),NV(10))
      GO TO 3
26    CALL CHANGE(F,AA,IA,I1,I2,NV(11),NV(12))
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHANGE(A,AA,IA,I1,I2,N,M)
      DIMENSION A(1),AA(1)
6     CONTINUE
103   FORMAT(10I5)
      IF(IA.EQ.'E') GO TO 1
      IF(IA.EQ.'C') GO TO 2
      IF(IA.EQ.'R') GO TO 3
      IF(IA.EQ.'M') GO TO 5
      IF(IA.EQ.'D') GO TO 7
      TYPE 100,IA
100   FORMAT(1X,A2,'  OPTION DOES NOT EXISTS - TRY E,C,R,D OR M')
      IER=1
      ACCEPT 102,IA
102   FORMAT(A1)
101   FORMAT(1X,A1)
      TYPE 101,IA
      GO TO 6
5     CONTINUE
      NM=N*M
      DO 10 I=1,NM
10    A(I)=AA(I)
      RETURN
7     CONTINUE
      DO 8 I=1,N
C   N=MAX(N,M)
      J=1+(N+1)*(I-1)
8     A(J)=AA(I)
      RETURN
3     J=-N+I1
      DO 31 II=1,M
      J=J+N
31    A(J)=AA(II)
      RETURN
2     JC=(I1-1)*N+1
      DO 11 I=1,N
      A(JC)=AA(I)
11    JC=JC+1
      RETURN
1     J=I1+(I2-1)*N
      A(J)=AA(1)
      RETURN
      END
```

```
      SUBROUTINE DSKRD(NMAT,A,B,C,NV,ITF)
      INTEGER R,REC,RECNUM,E,Y,F
      DIMENSION ITF(1),A(1),NV(1),B(1),C(1),RECOR(103),REC(2)
      EQUIVALENCE(NAM,REC(1)),(RECNUM,REC(2))
      CALL DEFINE FILE(1,3,NXT,'NAMES',0,0)
      CALL DEFINE FILE(2,103,NZ,'MATRIC',0,0)
      ITP=1
      NUM=1
1     TYPE 100,ITF(NUM)
100   FORMAT('  ENTER THE RECORD NAME FOR MATRIX <',A3,'>',2X,$)
      R=1
      ACCEPT 101,NAME
      IF(NAME.EQ.'SKIP.') RETURN
101   FORMAT(A5)
3     READ(1#R,END=90)REC
      IF(NAM.EQ.NAME) GO TO 10
      R=NXT
      GO TO 3
10    READ(2#RECNUM)RECOR
102   FORMAT(' MATRIX <',A5,'> FOUND IN REC #',I3)
      IF(ITP.EQ.1) TYPE 102,NAME,REC(2)
      GO TO (20,21,22),NUM
20    CALL ADISK($4,NUM,A,NV,RECOR)
21    CALL ADISK($4,NUM,B,NV,RECOR)
22    CALL ADISK($4,NUM,C,NV,RECOR)
4     IF(NUM-NMAT)1,1,5
90    TYPE 105,NAME
105   FORMAT('  RECORD WITH NAME <',A5,'> DOES NOT EXIST ')
      GO TO 1
5     CALL RELEAS(1)
      CALL RELEAS(2)
      RETURN
      END
```

```
      SUBROUTINE ADISK($,N,A,NV,R)
      DIMENSION A(1),NV(1),R(1)
      J=2*N-1
      NV(J)=R(2)
      NV(J+1)=R(3)
      NM=NV(J)*NV(J+1)
      DO 1 I=1,NM
1     A(I)=R(I+3)
      N=N+1
      RETURN 1
      END
```

```
      SUBROUTINE IOMT(A,N,M,IP,NTX,ITX)
      DIMENSION A(1),ITX(10)
      ID=IP
      IF(ID.LE.2) GO TO 6
      IF(ID.EQ.3) ID=1
      NM=N*M
      DO 1 I=1,NM
1     IF(A(I).GT.999.9.OR.A(I).LT.-99.9) ID=2
6     IB=1H
      J1=(NTX-1)/5+1
      TYPE 109,(ITX(K),K=1,J1)
109   FORMAT(10X,10A5)
      IF(ID.GT.0) GO TO 2
      CALL IM(A,N,M)
      IF(ID.EQ.0) RETURN
      ID = - ID
2     CONTINUE
      L1=1
      NM=N*M
      DO 8 I=1,NM
8     IF(A(I).GT.999.9.OR.A(I).LT.-99.9) ID=2
      M1=M
      IF(N.EQ.1) GO TO 5
      L1=(M-1)/(10/ID)+1
      IF(M*ID.GT.10) M1=10/ID
5     NM1=N*M1
      NM2=-NM1
      DO 4 L = 1,L1
      NM2=NM2+NM1
      NM3=NM2+NM1
      IF(L.EQ.L1) NM3=N*M
      DO 3 I = 1,N
      I1=I+NM2
      IF(ID.EQ.1)        TYPE 110,(A(J),J=I1,NM3,N)
      IF(ID.EQ.2)        TYPE 111,(A(J),J=I1,NM3,N)
3     CONTINUE
      TYPE 109,IB
4     CONTINUE
110   FORMAT(1X,10F6.2)
111   FORMAT(1X,5E12.5)
      RETURN
      END


      SUBROUTINE INTT(IT,I2,INTX,INT)
      DIMENSION IT(1)
      DIMENSION IN(10)
      DATA IN/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/
      INT=0
      IM=0
      IP=0
      DO 3 I=1,I2
      IF(IT(I).EQ.1H ) GO TO 3
      DO 2 J=1,10
2     IF(IT(I).EQ.IN(J)) GO TO 3
      IF(IT(I).EQ.1H-) IM=IM+1
      IF(IM.GT.1) GO TO 10
      IF(IT(I).EQ.1H-) GO TO 3
      IF(INTX.EQ.0) GO TO 10
      IF(IT(I).EQ.1H.) IP=IP+1
      IF(IT(I).NE.1H.) GO TO 10
      IF(IP.GT.1) GO TO 10
3     CONTINUE
      RETURN
10    INT=1
      RETURN
      END


      SUBROUTINE IM(A,N,M)
      DIMENSION IT(60),IF(4),IN(20),ID(20),JF(16),KF(10),A(1)
101   FORMAT(60A1)
      DO 10 IR = 1,N
9     CONTINUE
      IJ = IR-N
      L=1
      ACCEPT 101,IT
      DO 2 I = 1,60
      I1=60-I+1
      IF(IT(I1).NE.' ') GO TO 11
2     CONTINUE
11    CONTINUE
      IF(IT(I1).EQ.',') GO TO 12
      IT(I1+1) =','
12    CONTINUE
      IN(1)=-1
      DO 3 I = 1,60
      IF(IT(I).NE.',') GO TO 3
      L=L+1
      I1 = I-1
      IN(L) = I1
3     CONTINUE
104   FORMAT(' TOO MANY ELEMENTS IN THE ROW - TRY AGAIN')
      IF(L-1.LE.M) GO TO 8
      TYPE 104
      GO TO 9
8     CONTINUE
      M1=M+1
      DO 4 J = 2,M1
      I1=IN(J-1)+2
      I2=IN(J)
      ID(J-1) = I2-I1+1
      AN=0.
      IF(J.GT.L) GO TO 7
      IF(ID(J-1).LE.0) GO TO 7
      DO 5 I = 1,16
5     JF(I)=1H
      DO 6 I = I1,I2
      JF(-I+I1+16) = IT(-I+I2+I1)
6     CONTINUE
      CALL INTT(JF,16,1,INT)
      IF(INT.EQ.0) GO TO 1
      TYPE 105,JF
105   FORMAT('  I BEG YOUR PARDON  ',16A1)
      GO TO 9
1     CONTINUE
      ENCODE(16,107,KF) JF
107   FORMAT(16A1)
      DECODE(16,109,KF) AN
7     CONTINUE


      IJ=IJ+N
      A(IJ)=AN
109   FORMAT(F16.0)
4     CONTINUE
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OUTDAT(NM,NV,ITF,A,B,C,D,E,F)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),NV(1),ITF(1)
      IO='<'
      IC='>'
      TYPE 102,(IO,ITF(I),IC,I=1,NM)
      GO TO 5
6     TYPE 101
101   FORMAT(' FOR WRITING MATRICES ON TTY, LPT, DSK,OR NONE')
102   FORMAT(' AVAILABLE NAMES ARE ',6(A1,A3,A1))
5     CALL IFFM($1,$2,$3,$6,$4,'T','L','D','H','N',29,
     1 ' TYPE <T>,<L>,<D>,<N> OR HELP')
1     CALL OUTTTY(A,B,C,D,E,F,NV,NM,ITF)
      GO TO 5
2     CALL OUTLPT(A,B,C,D,E,F,NV,NM,ITF)
      GO TO 5
3     CALL OUTDSK(A,B,C,D,E,F,NV,NM,ITF)
      GO TO 5
4     RETURN
      END
```

```
      SUBROUTINE OUTTTY(A,B,C,D,E,F,NV,NM,ITF)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),NV(1),ITF(1),IT(10)
6     CONTINUE
      TYPE 102
102   FORMAT(' TYPE NAMES OF MATRICES YOU WANT TO TYPE ON TTY',
     1/' OR TYPE EITHER <N> OR <ALL>')
      CALL RCHAR(IT,1,NM)
      N=NM
      CALL IFALL($10,$7,NM,IT,ITF)
7     DO 1 I=1,N
      JJ=N-I+1
      IF(IT(JJ).EQ.'N') RETURN
1     IF(IT(JJ).NE.' ')GO TO 2
2     CONTINUE
      N=JJ
10    CONTINUE
      DO 3 I=1,N
      DO 4 J=1,NM
      JS=J
4     IF(IT(I).EQ.ITF(J)) GO TO 5
      TYPE 101,IT(I)
101   FORMAT(' MATRIX WITH NAME <',A3,'>  DOES NOT EXIST - TRY AGAIN')
      GO TO 6
5     CONTINUE
      GO TO (21,22,23,24,25,26),JS
26    CALL IOMT(F,NV(11),NV(12),3,3,ITF(6))
      GO TO 3
25    CALL IOMT(E,NV(9),NV(10),3,3,ITF(5))
      GO TO 3
24    CALL IOMT(D,NV(7),NV(8),3,3,ITF(4))
      GO TO 3
23    CALL IOMT(C,NV(5),NV(6),3,3,ITF(3))
      GO TO 3
22    CALL IOMT(B,NV(3),NV(4),3,3,ITF(2))
      GO TO 3
21    CALL IOMT(A,NV(1),NV(2),3,3,ITF(1))
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OUTLPT(A,B,C,D,E,F,NV,NM,ITF)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),NV(1),ITF(1),IT(10)
6     CONTINUE
      TYPE 102
102   FORMAT(' TYPE NAMES OF MATRICES YOU WANT TO PRINT ON LPT',
     1/' OR TYPE EITHER <N> OR <ALL>')
      CALL RCHAR(IT,1,NM)
      N=NM
      CALL IFALL($10,$7,NM,IT,ITF)
7     DO 1 I=1,N
      JJ=N-I+1
      IF(IT(JJ).EQ.'N') RETURN
1     IF(IT(JJ).NE.' ')GO TO 2
2     CONTINUE
      N=JJ
10    CONTINUE
      DO 3 I=1,N
      DO 4 J=1,NM
      JS=J
4     IF(IT(I).EQ.ITF(J)) GO TO 5
      TYPE 101,IT(I)
101   FORMAT(' MATRIX WITH NAME <',A3,'>  DOES NOT EXIST - TRY AGAIN')
      GO TO 6
5     CONTINUE
      GO TO (21,22,23,24,25,26),JS
26    CALL PRINT(F,NV(11),NV(12),2,1,ITF(6))
      GO TO 3
25    CALL PRINT(E,NV(9),NV(10),2,1,ITF(5))
      GO TO 3
24    CALL PRINT(D,NV(7),NV(8),2,1,ITF(4))
      GO TO 3
23    CALL PRINT(C,NV(5),NV(6),2,1,ITF(3))
      GO TO 3
22    CALL PRINT(B,NV(3),NV(4),2,1,ITF(2))
      GO TO 3
21    CALL PRINT(A,NV(1),NV(2),2,1,ITF(1))
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE OUTDSK(A,B,C,D,E,F,NV,NM,ITF)
      DIMENSION A(1),B(1),C(1),D(1),E(1),F(1),NV(1),ITF(1),IT(10)
6     CONTINUE
      TYPE 102
102   FORMAT(' TYPE NAMES OF MATRICES YOU WANT TO STORE ON DISK',
     1/' OR TYPE EITHER <N> OR <ALL>')
      CALL RCHAR(IT,1,NM)
      N=NM
      CALL IFALL($10,$7,NM,IT,ITF)
7     DO 1 I=1,N
      JJ=N-I+1
      IF(IT(JJ).EQ.'N') RETURN
1     IF(IT(JJ).NE.' ')GO TO 2
2     CONTINUE
      N=JJ
10    CONTINUE
      DO 3 I=1,N
      DO 4 J=1,NM
      JS=J
4     IF(IT(I).EQ.ITF(J)) GO TO 5
      TYPE 101,IT(I)
101   FORMAT(' MATRIX WITH NAME <',A3,'>  DOES NOT EXIST - TRY AGAIN')
      GO TO 6
5     CONTINUE
      GO TO (21,22,23,24,25,26),JS
26    CALL DSKWR(F,NV(11),NV(12),ITF(6))
      GO TO 3
25    CALL DSKWR(E,NV(9),NV(10),ITF(5))
      GO TO 3
24    CALL DSKWR(D,NV(7),NV(8),ITF(4))
      GO TO 3
23    CALL DSKWR(C,NV(5),NV(6),ITF(3))
      GO TO 3
22    CALL DSKWR(B,NV(3),NV(4),ITF(2))
      GO TO 3
21    CALL DSKWR(A,NV(1),NV(2),ITF(1))
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DSKWR(A,N,M,NAM1)
      INTEGER R,REC,RECNUM
      DIMENSION REC(2),RECOR(103),A(1)
      EQUIVALENCE(NAM,REC(1)),(RECNUM,REC(2))
      CALL DEFINE FILE(1,3,NXT,'NAMES',0,0)
      CALL DEFINE FILE(2,103,NZ,'MATRIC',0,0)
      ITP=1
      TYPE 105,NAM1
105   FORMAT(' ENTER RECORD NAME FOR MATRIX <',A3,'>',2X,$)
      ACCEPT 102,NAME
      GO TO 2
1     R=1
      RECNUM=0
10    READ(1#R,END=90)REC
      IF(NAM.EQ.NAME) GO TO 3
      R=NXT
      GO TO 10
90    READ(1#(R-1),END=91)REC
91    RECNUM=RECNUM+1
      WRITE(1#R)NAME,RECNUM
11    RECOR(1)=N*M
      RECOR(2)=N
      RECOR(3)=M
      J=4
      NN=N*M
106   FORMAT(' MATRIX <',A5,'> STORED IN REC #',I3)
      IF(ITP.EQ.1) TYPE 106,NAME,REC(2)
      DO 12 I=1,NN
      RECOR(J)=A(I)
      J=J+1
12    CONTINUE
      WRITE(2#RECNUM)RECOR
      GO TO 20
3     TYPE 100,NAM
      GO TO 5
100   FORMAT('  RECORD WITH NAME <',A5,'> ALREADY EXISTS')
104   FORMAT(' TO ENTER NEW NAME,OVERWRITE OLD MATRIX, OR SKIP')
6     TYPE 104
5     CALL IFFM($4,$11,$6,$20,$20,'N','O','W','S','S',25,
     1' TYPE <N>,<O>,<S> OR HELP')
4     TYPE 101
101   FORMAT('  ENTER NEW NAME  ',$)
      ACCEPT 102,NAME
102   FORMAT(A5)
2     IF(NAME.NE.' ') GO TO 1
      TYPE 103,NAME
103   FORMAT(' NAME <',A5,'>  IS NOT PERMITTED')
      GO TO 4
20    CALL RELEAS(1)
      CALL RELEAS(2)
      RETURN
      END
```

```
      SUBROUTINE PRINT(A,N,M,ID,NTX,ITX)
      DIMENSION A(1),ITX(10)
      IB=1H
      J1=(NTX-1)/5+1
      PRINT 109,(ITX(K),K=1,J1)
109   FORMAT(10X,10A5)
      IF(ID.GT.0) GO TO 2
      IF(ID.EQ.0) RETURN
      ID = - ID
2     CONTINUE
      L1=1
      M1=M
      IF(N.EQ.1) GO TO 5
      L1=(M-1)/(10/ID)+1
      IF(M*ID.GT.10) M1=10/ID
5     NM1=N*M1
      NM2=-NM1
      DO 4 L = 1,L1
      NM2=NM2+NM1
      NM3=NM2+NM1
      IF(L.EQ.L1) NM3=N*M
      DO 3 I = 1,N
      I1=I+NM2
      IF(ID.EQ.1)       PRINT 110,(A(J),J=I1,NM3,N)
      IF(ID.EQ.2)       PRINT 111,(A(J),J=I1,NM3,N)
3     CONTINUE
      PRINT 109,IB
4     CONTINUE
110   FORMAT(1X,10F6.2)
111   FORMAT(1X,5E12.5)
      RETURN
      END
```

```
      SUBROUTINE CALDIM(NVI,NV,IM,N,M,K,L)
      DIMENSION NVI(1),NV(1)
      DO 1 I=1,IM
      IF(NVI(I).EQ.1) NV(I)=N
      IF(NVI(I).EQ.2) NV(I)=M
      IF(NVI(I).EQ.3) NV(I)=K
      IF(NVI(I).EQ.4) NV(I)=L
      IF(NVI(I).GT.4.OR.NVI(I).LT.1) TYPE 101,I
101   FORMAT('  NVI(',I2,'  IS OUT OF BOUNDS CORRECT NVI(I) OR IM')
1     CONTINUE
      RETURN
      END


      SUBROUTINE TYPDIM(NM,NV,ITF)
      DIMENSION ITF(1),NV(1)
      IB=' '
      TYPE 102,IB
      IO='<'
      TYPE 101,(ITF(I),I=1,NM)
      JM=NM*2
      TYPE 102,(IO,NV(I),NV(I+1),I=1,JM,2)
101   FORMAT(1X,6(2X,A3,3X))
102   FORMAT(1X,6(A1,I2,',',I2,'>',1X))
      TYPE 102,IB
      RETURN
      END


      SUBROUTINE IFALL($,$,N,IT,ITF)
      DIMENSION IT(1),ITF(1)
      IF(IT(1).NE.'ALL') RETURN 2
      DO 1 I=1,N
1     IT(I)=ITF(I)
      RETURN 1
      END
```

```
      SUBROUTINE RCHAR(JT,N,M)
      DIMENSION IT(60),IF(4),IN(20),ID(20),JF(5),KF(10),JT(1)
101   FORMAT(60A1)
      DO 10 IR = 1,N
9     CONTINUE
      IJ = IR-N
      L=1
      ACCEPT 101,IT
      DO 2 I = 1,60
      I1=60-I+1
      IF(IT(I1).NE.' ') GO TO 11
2     CONTINUE
11    CONTINUE
      IF(IT(I1).EQ.',') GO TO 12
      IT(I1+1) =','
12    CONTINUE
      IN(1)=-1
      DO 3 I = 1,60
      IF(IT(I).NE.',') GO TO 3
      L=L+1
      I1 = I-1
      IN(L) = I1
3     CONTINUE
104   FORMAT(' TOO MANY ELEMENTS IN THE ROW - TRY AGAIN')
      IF(L-1.LE.M) GO TO 8
      TYPE 104
      GO TO 9
8     CONTINUE
      M1=M+1
      DO 4 J = 2,M1
      I1=IN(J-1)+2
      I2=IN(J)
      ID(J-1) = I2-I1+1
      AN=0.
      IF(J.GT.L) GO TO 7
      IF(ID(J-1).LE.0) GO TO 7
      DO 5 I = 1,5
5     JF(I)=1H
      DO 6 I = I1,I2
      JF(I2-I+1)=IT(-I+I2+I1)
6     CONTINUE
      ENCODE(5,107,KF) JF
107   FORMAT(5A1)
      DECODE(5,109,KF) JAN
7     CONTINUE
      IJ=IJ+N
      JT(IJ)=JAN
109   FORMAT(A5)
4     CONTINUE
10    CONTINUE
      DO 13 I=L,M
13    JT(I)=5H
      RETURN
      END
```

```
      SUBROUTINE HLP(II)
      GO TO(1,2,3,4,5,6,7,8,9,10,11,12),II
1     TYPE 101
101   FORMAT(//' IN CASE OF USING DISK DATA FILE IT IS NECESSARY'/
     1/' TO ASSIGN DSK:2, IT MAY BE DONE BY TYPING:'/
     2/' <CTRL>C; <ASS DSK:2>,<CR> AND <CONT>,<CR>'///)
      RETURN
2     TYPE 102
102   FORMAT(/' TO CHANGE INTEGERS (MATRIX DIMENSIONS , ETC.);'/
     1'           ELEMENTS OF MATRICES; OR TO'/
     2'           CONTINUE'/)
      RETURN
3     TYPE 103
103   FORMAT(/'  TO CHANGE INTEGERS (MATRIX DIMENSIONS, ETC.);'/
     1'            ELEMENTS OF MATRICES OR TO '/
     2'            RETURN TO LINSYS'/)
      RETURN
4     TYPE 104
104   FORMAT(/' TO OBTAIN JORDAN FORM; TO APPLY'/
     1,'           ARBITRARY STATE TRANSFORMATION; OR'/
     2,'           NONE')
      RETURN
5     TYPE 105
105   FORMAT(/' TO OBTAIN DISCRETE TRANSF.FUN.MATRIX'
     1' (STATE TRANSIT.MAT);'/
     1'           CONTIN.  TRANSF.FUN.MATRIX; OR'/
     2'           NONE')
      RETURN
6     TYPE 106
106   FORMAT(/' TO CHANGE INTEGERS (MATRIX DIMENSIONS, ETC.);'/
     1'           ELEMENTS OF MATRICES; TO OBTAIN NEW'/
     2'           PLOTS; OR TO '/
     3'           RETURN TO LINSYS'/)
      RETURN
7     TYPE 107
107   FORMAT(/' TO CHANGE ELEMENTS IN NO,E,X0 AND TT;'/
     1'           INDICES; OR '/
     2'           NONE'/)
      RETURN
8     RETURN
9     RETURN
10    RETURN
11    RETURN
12    RETURN
      END
```

```
C       LOC.F4
      SUBROUTINE LOC(I,J,IR,N,M,MS)
      IX=I
      JX=J
      IF(MS-1) 10,20,30
10    IRX=N*(JX-1)+IX
      GO TO 36
20    IF(IX-JX) 22,24,24
22    IRX=IX+(JX*JX-JX)/2
      GO TO 36
24    IRX=JX+(IX*IX-IX)/2
      GO TO 36
30    IRX=0
      IF(IX-JX) 36,32,36
32    IRX=IX
36    IR=IRX
      RETURN
      END
```

```
      SUBROUTINE IFF($,$,NT,IT)
      DIMENSION IT(10),JY(10)
      K=(NT-1)/5+1
1     TYPE 101,(IT(I),I=1,K)
      IE=0
      ACCEPT 100,JY
      DO 2 I=1,10
2     IF(JY(I).NE.' ') GO TO 3
      GO TO 4
3     IY=JY(I)
      IF(IY.EQ.'Y') IE=1
      IF(IY.EQ.'N') IE=2
      IF(IE.NE.0) RETURN IE
4     TYPE 102,IY
      GO TO 1
100   FORMAT(10A1)
101   FORMAT(1X,10A5)
102   FORMAT(' ?',A1,'? - TRY AGAIN')
      END
```

```
      SUBROUTINE IFFM($,$,$,$,$,IS1,IS2,IS3,IS4,IS5,NT,IT)
      DIMENSION IT(10),JY(10)
      K=(NT-1)/5+1
1     TYPE 101,(IT(I),I=1,K)
      IE=0
      ACCEPT 100,JY
      DO 2 I=1,10
2     IF(JY(I).NE.' ') GO TO 3
      GO TO 4
3     IY=JY(I)
      IF(IY.EQ.IS1) IE=1
      IF(IY.EQ.IS2) IE=2
      IF(IY.EQ.IS3) IE=3
      IF(IY.EQ.IS4) IE=4
      IF(IY.EQ.IS5) IE=5
      IF(IE.NE.0) RETURN IE
4     TYPE 102,IY
      GO TO 1
100   FORMAT(10A1)
101   FORMAT(1X,10A5)
102   FORMAT(' ?',A1,'? - TRY AGAIN')
      END
```

```
      SUBROUTINE RIC(A,B,Q,R,AK,EPS,N,M,IT,IJ,C)
C                SOLVES RICCATI EQUATION
C              A*AK + AT*AK - AK*S*AK + Q = 0
C                S = B*(R**-1)*BT
      DIMENSION A(1),B(1),Q(1),R(1),AK(1),C(1)
      COMMON L1(10),M1(10),EV(100),EVVV(100),AI(100),DK(100)
     1,E(100),S(100),UC(100),SS(100),EV2V(100)
      ITR=0
      ITRM=50
      EP1=10.**(-12)
      F05=0.05
      JST=0
      KST=0
      NN=N*N
100   FORMAT(10I3)
      IF(M-1) 10,10,11
10    IF(R(1).LT.EP1) GO TO 14
      R(1)=1./R(1)
      D=R(1)
      GO TO 13
11    CALL MINV(R,M,D,L1,M1)
13    IF(D.GT.EP1) GO TO 12
14    TYPE 108,EP1
108   FORMAT(/'  DET. OF R IS LESS THAN ',E12.3/)
      GO TO 4
12    CONTINUE
      ISTAB=0
      CALL GMPRD(B,R,C,N,M,M)
      CALL GMTRA(B,E,N,M)
      CALL GMPRD(C,E,S,N,M,N)
      CALL MCPY(S,SS,N,N,0)
15    IF(JST.NE.0) CALL IOMT(S,N,N,1,1,'S')
      ITR=ITR+1
      IF(ITR.GE.ITRM) GO TO 6
      DO 1 IJ=1,IT
      CALL GMPRD(S,AK,C,N,N,N)
      CALL GMSUB(A,C,C,N,N)
      IF(IJ.GT.1) GO TO 3
      CALL STT(C,N,E,EV,IND)
      IF(IND.EQ.1) GO TO 3
      IF(ISTAB.EQ.1) F05=F05*1.5
      IF(ISTAB.EQ.1) GO TO 7
      ICC=0
      ES=5.
      CC=5.
5     CC=CC*2.
      ES=ES*2.
      CALL STABK(A,UC,AK,S,N,CC,ES,C,IND)
      ICC=ICC+1
      IF(ICC.GT.6) GO TO 6
      ISTAB=1
      IF(IND.EQ.0) GO TO 5
      CALL MCPY(UC,S,N,N,0)
3     CALL MCPY(C,AI,N,N,0)
      CALL TRC(A,S,Q,AK,N,E,UC,DK)
      ITM=IT
      CALL SMPY(E,-1.,E,N,N,0)
      CALL LYAP(C,E,DK,1.,EPS,N,ITM,0)
      IF(ITM.NE.IT) TYPE 100,N,IT,ITM
      CALL GMADD(AK,DK,AK,N,N)
      CALL GMPRD(DK,DK,EV2,1,NN,1)
      EV2=SQRT(EV2)/NN
      CALL GMPRD(AK,AK,AKK,1,NN,1)
      AKK=SQRT(AKK)/NN
      EV2=EV2/AKK
      EV2V(IJ)=EV2
      IF(EV2.LE.EPS) GO TO 2
1     CONTINUE
2     CONTINUE
      CALL MCPY(AI,C,N,N,0)
107   FORMAT(I4,'   ITERATIONS OUT OF ',I4,2E15.3)
C       JST=1
      IF(JST.NE.0) TYPE 107,IJ,IT,FF,ES
      IF(ISTAB.EQ.0) CALL IOMT(EV2V,1,IJ,2,3,'ERR')
      IF(ISTAB.EQ.0) RETURN
7     FF=F05*IJ
      IF(FF.GE.1.) FF=0.8
      ES=ES*FF
      IF(IJ.LE.2) GO TO 12
      CALL SCLA(AI,0.,N,N,0)
      CALL DCLA(AI,ES,N,0)
      CALL GMADD(SS,AI,S,N,N)
      GO TO 15
6     TYPE 101
101   FORMAT(' SYSTEM NOT STABILIZABLE')
4     IJ=0
      RETURN
      END


      SUBROUTINE LYAP(A,B,X,Q,EPS,N,ITERM,ITRANS)
C               SOLVES LYAPUNOV EQUATION
C               AT*X + X*A + B = 0; FOR ITRANS = 0

C               A *X + X*AT+ B = 0; FOR ITRANS = 0
      DIMENSION A(1),B(1),X(1)
      COMMON L(10),LL(10),U(100),V(100)
      IF(ITRANS.EQ.1) GO TO 2
      CALL GMTRA(A,U,N,N)
      CALL MCPY(U,A,N,N,0)
2     CONTINUE
      CALL SCLA(U,0.,N,N,0)
      CALL DCLA(U,Q,N,0)
      CALL GMADD(U,A,V,N,N)
      CALL GMSUB(U,A,U,N,N)
      CALL MINV(U,N,DET,L,LL)
      CALL GMPRD(U,V,A,N,N,N)
      CALL GMPRD(U,B,V,N,N,N)
      CALL GMTRA(U,X,N,N)
      CALL GMPRD(U,B,V,N,N,N)
      CALL GMPRD(V,X,U,N,N,N)
      CALL SMPY(U,2.*Q,X,N,N,0)
      CALL GMTRA(A,U,N,N)
      CALL GMPRD(A,X,B,N,N,N)
      CALL GMPRD(B,U,V,N,N,N)
      CALL GMADD(X,V,X,N,N)
      DO 200 ITER=1,ITERM
      CALL GMPRD(A,A,V,N,N,N)
      CALL GMTRA(V,U,N,N)
      CALL GMPRD(V,X,B,N,N,N)
      CALL GMPRD(B,U,A,N,N,N)
      CALL GMADD(A,X,A,N,N)
      CALL TEST(X,A,N,EPS,U,IL)
      L(1)=ITER
      IF(IL.EQ.1) RETURN
      CALL MCPY(V,A,N,N,0)
200   CONTINUE
      TYPE 101,L(1)
101   FORMAT(  '    LYAP NO. OF ITER=',I10)
      ITERM=0
      RETURN
      END
```

```
      SUBROUTINE TRC(A,S,Q,AK,N,Y,C,D)
      DIMENSION A(1),S(1),Q(1),AK(1),Y(1),C(1),D(1)
      CALL GMPRD(AK,S,C,N,N,N)
      CALL GMPRD(C,AK,Y,N,N,N)
      CALL GMSUB(Y,Q,Y,N,N)
      CALL GMPRD(AK,A,C,N,N,N)
      CALL GMSUB(Y,C,Y,N,N)
      CALL GMTRA(C,D,N,N)
      CALL GMSUB(Y,D,Y,N,N)
      RETURN
      END
```

```
      SUBROUTINE STT(A,N,RR,RI,IND)
      DIMENSION A(1),RR(1),RI(1)
      DIMENSION C(100)
      COMMON L1(10),M1(10),D(100)
      IHSBG=1
      IF(IHSBG.EQ.0) GO TO 3
      CALL MCPY(A,C,N,N,0)
      CALL HSBG(N,C,N)
      CALL ATEIG(N,C,RR,RI,L1,N)
      GO TO 4
3     CALL CHEQ(A,N,D,RR,RI)
      D(N+1)=1.
      CALL POLRT(D,C,N,RR,RI,IE)
      IF(IE.NE.0) TYPE 100,IE
C       IF(IE.NE.0) CALL POLRT1(D,C,N,RR,RI,IE)
100   FORMAT(10I3)
4     IND=1
      DO 1 I=1,N
1     IF(RR(I).GE.0.) IND=0
      RETURN
      END
```

```
      SUBROUTINE TEST(AK,X,N,EPS,P1,IEK)
      DIMENSION AK(1),X(1),P1(1)
      IEK=0
      NN=N*N
      CALL GMSUB(AK,X,P1,N,N)
      CALL GMPRD(P1,P1,EP,1,NN,1)
      EP=SQRT(EP)/NN
      CALL GMPRD(AK,AK,EK,1,NN,1)
      EK=SQRT(EK)/NN
      EP=EP/EK
      IF(EP.LE.EPS) IEK=1
      CALL MCPY(X,AK,N,N,0)
      RETURN
      END
```

```
      SUBROUTINE STABK(A,B,AK,S,N,CC,ES,C,IND)
      DIMENSION A(1),B(1),AK(1),S(1),C(1)
      COMMON TT(20),AI(100),RR(10),RI(10)
      IST=0
      CALL SCLA(AI,0.,N,N,0)
      CALL DCLA(AI,1.,N,0)
      CALL SMPY(AI,ES,C,N,N,0)
      CALL GMADD(C,S,C,N,N)
      CALL MCPY(C,B,N,N,0)
      CALL JACC(A,C,N,AK,CC)
      CALL GMPRO(C,AK,AI,N,N,N)
      CALL GMSUB(A,AI,AI,N,N)
      CALL STT(AI,N,RR,RI,IND)
      IF(IST.EQ.0) GO TO 7
      CALL IOMT(RR,1,N,2,2,'RR')
      CALL IOMT(AK,N,N,2,6,'K-TEMP')
7     CALL MCPY(AI,C,N,N,0)
      RETURN
      END
```

```
      SUBROUTINE JACC(A,S,N,AKS,C)
      DIMENSION A(1),S(1),AKS(1)
      COMMON AR(100),SR(100)
100   FORMAT(10I3)
      DO 2 K =1,N
      L=0
      DO 1 I=K,N
      IA=I-K+1
      JA=I+(K-1)*N
      AR(IA)=A(JA)
      DO 1 J=K,N
      L=L+1
      IS=J+(I-1)*N
1     SR(L)=S(IS)
      IS=N-K+1
      AR(1)=AR(1)+C
      IF(K.EQ.N) GO TO 10
      CALL SIMQ(SR,AR,IS,IE)
      IF(IE.EQ.1) TYPE 100,IE,IE,N
      GO TO 11
10    AR(1)=AR(1)/SR(1)
11    CONTINUE
      DO 3 I=1,IS
      K1=1+(K-1)*(N+1)
      I1=K1+I-1
      I2=K1+(I-1)*N
      AKS(I1)=AR(I)
      AKS(I2)=AR(I)
3     CONTINUE
2     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PPL(A,B,RR,RI,AD,N,M,EPS,D,P1,AF,ICO,IND)
      DIMENSION A(1),B(1),RR(1),RI(1),AD(1),D(1),P1(1),AF(1)
      COMMON L1(10),M1(10),A1(100),B1(100),H(100),H1(100),Q(100)
     2,NV(10),NZ(10),LV(10)
      JST=0
100   FORMAT(10I3)
      IF(IND.EQ.1HC) GO TO 14
      CALL GMTRA(A,H,N,N)
      CALL MCPY(H,A,N,N,0)
      CALL GMTRA(B,H,M,N)
      CALL MCPY(H,B,N,M,0)
14    CONTINUE
      CALL COIN2(A,B,N,M,NZ,NV,LV,'C',EPS,H)
      IF(NV(M+1).NE.N) GO TO 2
104   FORMAT(' CONTROLLABILITY/OBSERVABILITY INDICES',10I3)
      CALL MCPY(A,A1,N,N,0)
      TYPE 104,(NZ(I),I=1,M)
      CALL MCPY(B,B1,N,M,0)
      CALL TRANS(A1,B1,D,H,H1,N,M,1,1)
      CALL GMTRA(A1,D,N,N)
      JP=N*N+1
      CALL FRT(D,LV,NV,NV(M),N,M,P1,AF,Q)
      CALL MCPY(P1(JP),P1,N,N,0)
      CALL GMTRA(P1,AF,N,N)
      DO 5 I=1,N
      IA=I-N
      IB=1-I
      DO 5 J=1,N
      IA=IA+N
      IB=IB+N
5     Q(IB)=AF(IA)
      L=0
      DO 10 I=1,N
      DO 11 J=1,M
      IQ=I+(J-1)*N
11    IF(ABS(Q(IQ)).GT.EPS) GO TO 12
      GO TO 13
12    L=L+1
      LV(L)=I
13    CONTINUE
10    CONTINUE
      CALL TRANS(A1,B1,D,P1,Q,N,M,1,2)
      TYPE 106,(NV(I),I=1,M)
106   FORMAT(' ORDERED CONTROLLABILITY/OBSERVABILITY INDICES',10I3)
      CALL RED(A1,N,M,N,LV,D)
      CALL SCLA(AF,0.,M,N,0)
      CALL SCLA(AD,0.,M,N,0)
      CALL FAD(NV,M,N,L1,RR,RI,AF,AD)
      IF(JST.NE.0) CALL IOMT(AD,M,N,1,16,'M ROWS OF A2-DES')
      CALL GMSUB(D,AD,D,M,N)
      CALL RED(B1,N,M,M,LV,AF)
      CALL MINV(AF,M,DD,L1,M1)
      IF(ABS(DD).LE.EPS) CALL IOMT(DD,1,1,2,6,'DET=DD')
      CALL GMPRD(AF,D,P1,M,M,N)
      CALL MCPY(P1,AF,M,N,0)
      CALL GMPRD(Q,H1,P1,N,N,N)
      CALL GMPRD(AF,P1,D,M,N,N)
      IF(JST.NE.0) CALL IOMT(P1,N,N,1,3,'T-T')
      IF(JST.NE.0) CALL IOMT(D,M,N,1,6,'FEED-R')
      CALL GMPRD(B,D,P1,N,M,N)
      CALL SMPY(D,-1.,D,M,N,0)
      CALL GMSUB(A,P1,AF,N,N)
      IF(JST.NE.0) CALL IOMT(AF,N,N,1,3,'A-F')
103   FORMAT(' CLOSED LOOP EIG.VALUES')
      JPL=1
      CALL EGVP(AF,N,RR,RI,JPL)
      ICO=0
      IF(IND.EQ.1HC) RETURN
      CALL GMTRA(D,Q,M,N)
      CALL MCPY(Q,D,N,M,0)
      CALL GMTRA(AF,Q,N,N)
      CALL MCPY(Q,AF,N,N,0)
      CALL GMTRA(A,Q,N,N)
      CALL MCPY(Q,A,N,N,0)
      CALL GMTRA(B,Q,N,M)
      CALL MCPY(Q,B,M,N,0)
      RETURN
2     ICO=1
      RETURN
      END
```

```
      SUBROUTINE COIN2(A,B,N,M,NZ,NV,LV,IND,EPS,H)
      DIMENSION A(1),B(1),NZ(1),NV(1),LV(1),H(1)
      DIMENSION C(100),D(100),CP(100)
      IST=0
      IF(IND.EQ.1HC) GO TO 1
      CALL GMTRA(A,C,N,N)
      CALL MCPY(C,A,N,N,0)
      CALL GMTRA(B,C,M,N)
      CALL MCPY(C,B,N,M,0)
1     MM=0
      NV(M+1)=0
      K=M
      DO 2 J=1,M
2     NZ(J)=0
C       CALL RANK1(B,M,N,JR,EPS,CC,FM,D)
      CALL RANK2(B,M,DD,N,M,IR,EPS)
      IF(IR.EQ.0) GO TO 7
      CALL MCPY(B,C,N,M,0)
      CALL MCPY(B,H,N,M,0)
      I=1
3     I=I+1
C       IF(K.EQ.N) GO TO 10
      CALL GMPRD(A,C,D,N,N,M)
      CALL MCPY(D,C,N,M,0)
      DO 4 J=1,M
      IF(NZ(J).NE.0) GO TO 4
      IC=1+(J-1)*N
      IH=K*N+1
      CALL MCPY(C(IC),H(IH),1,N,0)
C       CALL RANK1(H,K+1,N,JR,EPS,CC,FM,D)
C       K1=K+1
      CALL RANK2(H,D,DD,N,K+1,IR,EPS)
      IF(IST.EQ.0) GO TO 8
C       D1=FM(K1)
      TYPE 102,DD,K,IR,I,J
102   FORMAT(' DET =', E10.3,5I5)
      ACCEPT 100,IR
8     IF(IR.EQ.0) GO TO 5
      K=K+1
      GO TO 4
5     NZ(J)=I-1
      MM=MM+1
      LV(MM)=K-M+MM
      NV(MM)=I-1
      IF(MM.EQ.M) GO TO 10
4     CONTINUE
      GO TO 3
10    NV(M+1)=K
      IF(K.GE.N) GO TO 6
      IF(IND.EQ.1HC) TYPE 105
      IF(IND.EQ.1HO) TYPE 106
6     CONTINUE
      IF(IND.EQ.1HC) RETURN
      CALL GMTRA(A,C,N,N,0)
      CALL MCPY(C,A,N,N,0)
      CALL GMTRA(B,C,N,M)
      CALL MCPY(C,B,M,N,0)
      CALL GMTRA(H,C,N,K)
      CALL MCPY(C,H,K,N,0)
100   FORMAT(20I3)
      RETURN
7     IF(IND.EQ.1HC) TYPE 103
      IF(IND.EQ.1HO) TYPE 104
103   FORMAT(' MATRIX B/C IS NOT OF FULL RANK')
104   FORMAT(' MATRIX B/C IS NOT OF FULL RANK')
105   FORMAT(' SYSTEM IS NOT CONTROLLABLE/OBSERVABLE')
106   FORMAT(' SYSTEM IS NOT CONTROLLABLE/OBSERVABLE')
      RETURN
      END


      SUBROUTINE RED(A,N,M,K,LV,B)
      DIMENSION A(1),B(1),LV(1)
      DO 1 I=1,M
      L=I-M
      I1=LV(I)-N
      DO 1 J=1,K
      L=L+M
      I1=I1+N
1     B(L)=A(I1)
      RETURN
      END


      SUBROUTINE EVCHE(RR,RI,N,Z,IZ)
      DIMENSION RR(1),RI(1),Z(1),X(10),Y(10)
      CALL SCLA(X,0.,1,10,0)
      CALL SCLA(Y,0.,1,10,0)
      EPS=0.001
      IX=1
      X(1)=1.
      I=0
1     I=I+1
      IF(I.GT.N) RETURN
      IF(ABS(RI(I)).GT.EPS) GO TO 2
      IY=2
      Y(1)=-RR(I)
      Y(2)=1.
      GO TO 3
2     IY=3
      Y(3)=1.
      Y(2)=-2.*RR(I)
      Y(1)=RR(I)*RR(I)+RI(I)*RI(I)
      I=I+1
3     CALL PMPY(Z,IZ,Y,IY,X,IX)
      IX=IZ
      CALL MCPY(Z,X,1,IZ,0)
      GO TO 1
      END
```

```
      SUBROUTINE FAD(NV,M,N,LL,RR,RI,COEF,P)
      DIMENSION NV(1),RR(1),RI(1),COEF(1),P(1),LL(1)
      IST=0
      L=0
      NVM=NV(M)
      DO 6 I=1,NVM
      DO 6 J=1,M
      J1=M-J+1
      IF(NV(J1).LT.I) GO TO 6
      L=L+1
      LL(L)=J1
12    CONTINUE
      IF(IST.EQ.1) TYPE 100,I,J,J1,L,NV(J1),LL(L)
100   FORMAT(20I3)
6     CONTINUE
      GO TO 10
C       CALL IFF($10,$11,26,'EG.OR M*N MATRIX - Y OR N')
10    CONTINUE
      IR=1
      I1=NV(1)
      EPS=.001
      NVM=NV(M)
4     DO 1 I=1,M
      IF(ABS(RI(I1)).LE.EPS) GO TO 2
      I2=I1+1
      DRR=RR(I1)-RR(I2)
      IF(ABS(DRR).GT.EPS) GO TO 2
      DRI=RI(I1)+RI(I2)
      IF(ABS(DRI).GT.EPS) GO TO 2
      TYPE 101
101   FORMAT(' REARANGE EIG.VAL. IN ACCORDANCE WITH CONT.IND.')
      TYPE 100,(NV(J),J=1,M)
      GO TO 4
2     CONTINUE
      CALL EVCHE(RR(IR),RI(IR),NV(I),P,IP)
      NVI=NV(I)
      DO 5 J=1,NVI
      L=I+(J-1)*M
      COEF(L)=-P(J)
5     CONTINUE
      IF(IST.NE.0) TYPE 100,I,I1,I2,IR,IP
      IR=IR+NV(I)
      I1=I1+NV(I+1)
1     CONTINUE
      CALL SCLA(P,0.,M,N,0)
      DO 3 I=1,M
      L=0
      DO 3 J=1,N
      IF(LL(J).NE.I) GO TO 3
      L=L+1
      IA=I+(J-1)*M
      IC=I+(L-1)*M
      P(IA)=COEF(IC)
      IF(IST.NE.0) TYPE 100,I,J,L,LL(J),IA,IC
3     CONTINUE
      IF(IST.NE.0) CALL IOMT(P,M,N,1,2,'P1')
      NV(M+1)=0
      I1=1
13    DO 7 I=I1,M
      IF(NV(I).EQ.NV(I+1))GO TO 7
      I2=I
      GO TO 8
7     CONTINUE
8     IF(I1.EQ.I2) GO TO 14
      I4=(I1+I2)/2
      DO 9 I=I1,I4
      I3=I2-I+I1
      DO 9 J=1,N
      L1=I+(J-1)*M
      L2=I3+(J-1)*M
      TMP=P(L1)
      P(L1)=P(L2)
      P(L2)=TMP
9     CONTINUE
14    CONTINUE
      I1=I2+1
      IF(I1.LT.M) GO TO 13
      IF(IST.NE.0) CALL IOMT(P,M,N,1,2,'P2')
      RETURN
11    CALL IOMT(P,M,N,-1,5,'A-DES')
      RETURN
      END


      SUBROUTINE FRT(A,L,NN,NM,N,IQ,T,TZ,B)
      DIMENSION A(1),L(1),NN(1),T(1),TZ(1),B(1)
      IT = 1-N
      DO 6 I = 1,IQ
      CALL SMPY (B,0.,B,1,N,0)
      B(L(I)) = 1.
      IT = IT+N
6     CALL MCPY (B,TZ(IT),1,N,0)
      ITT = 2*N+1
      DO 10 I = 1,NM
      DO 11 J = 1,IQ
      IF((I-1).LE.NN(J)) ITT = ITT-1
11    CONTINUE
      JTT = (ITT-1)*N+1
      KTT = ITT-N
      CALL MCPY (TZ,T(JTT),N,IQ,0)
      CALL GMPRD (A,TZ,B,N,N,IQ)
      CALL MCPY (B,TZ,N,IQ,0)
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE RANKR(A,B,RR,RI,N,M,EPS,IR,IND)
      DIMENSION A(1),B(1),RR(1),RI(1),IRV(1)
     1,AI(100),A12(100),A21(100),AT(100)
      IF(IND.EQ.'C') GO TO 5
      CALL GMTRA(A,AI,N,N)
      CALL MCPY(AI,A,N,N,0)
      CALL GMTRA(B,AI,M,N)
      CALL MCPY(AI,B,N,M,0)
5     CONTINUE
      ITY=0
      NM=N+M
      IR=N
      IA=N*N+1
      I=0
2     I=I+1
      CALL SCLA(AI,0.,N,N,0)
      CALL DCLA(AI,RR(I),N,0)
      CALL GMSUB(AI,A,AI,N,N)
      ID=1
      CALL MCPY(B,AI(IA),N,M,0)
      N2=N
      N3=NM
      CALL MCPY(AI,AT,N2,N3,0)
      IF(ITY.NE.0) CALL IOMT(AT,N2,N3,3,5,'S=A/B')
      IF(ABS(RI(I)).LE.0.00001) GO TO 3
      ID=2
      I=I+1
      CALL SCLA(A12,0.,N,NM,0)
      CALL DCLA(A12,RI(I),N,0)
      CALL SMPY(A12,-1.,A21,N,N,0)
      CALL JOIN(AI,A12,A21,AI,N,NM,N,NM,N2,N3,AT)
3     CALL GMTRA(AT,AI,N2,N3)
      IF(ITY.NE.0) CALL IOMT(AI,N3,N2,3,2,'AT')
      CALL RANK2(AI,AT,DD,N3,N2,IRR,EPS)
      IF(ITY.NE.0) TYPE 100,IRR,N
      IF(IRR.GE.N) GO TO 4
      IR=0
      IF(IND.EQ.'C') TYPE 102,RR(I),RI(I)
      IF(IND.EQ.'O') TYPE 103,RR(I),RI(I)
102   FORMAT(' MODE  ',E10.3,'  +/- J  ',E10.3,'  NOT CONTROLLABLE')
103   FORMAT(' MODE  ',E10.3,'  +/- J  ',E10.3,'  NOT OBSERVABLE')
4     CONTINUE
100   FORMAT(10I3)
101   FORMAT(' DET= ',2E12.3)
      IF(I.EQ.N) GO TO 6
      GO TO 2
6     IF(IND.EQ.'C') RETURN
      CALL GMTRA(A,AI,N,N)
      CALL MCPY(AI,A,N,N,0)
      CALL GMTRA(B,AI,N,M)
      CALL MCPY(AI,B,M,N,0)
      RETURN
      END
```

```
      SUBROUTINE JFORM(A1,B1,C1,N,M,K,T,TI)
      DIMENSION A1(1),B1(1),C1(1),T(1),TI(1)
      COMMON L1(20),P1(100),P2(100),Q(100),C(100)
100   FORMAT(10I3)
      EPS=0.01
      DO 4 I=1,N
      KK=2*I+1
      DO 2 J=1,N
      P1(J)=RAN(KK)
      IF(I.EQ.1) P1(J)=1.
2     CONTINUE
      CALL COMF(A1,P1,N,1HO,T,TI)
      D=DET(T,N)
      IF(ABS(D).GT.EPS) GO TO 3
4     CONTINUE
      CALL IOMT(D,1,1,2,6,'DET=TC')
3     CONTINUE
      CALL CHEQ(A1,N,Q,P1,P2)
      Q(N+1)=1
      DO 1 I=1,N
      IC=N-I+1
1     C(IC)=-Q(I)
      CALL MCPY(A1,Q,N,N,0)
      CALL HSBG(N,Q,N)
      CALL ATEIG(N,Q,P2,TI,L1,N)
      CALL ORD(P2,TI,N)
      CALL EGT(C,P2,TI,N,Q)
      CALL GMPRD(T,Q,C,N,N,N)
      INORM=0
      IF(INORM.EQ.0) GO TO 13
      DO 10 I=1,N
      L=(I-1)*N+1
      AB=ABS(C(L))
      DO 11 J=2,N
      L=L+1
      IF(AB.GE.ABS(C(L))) GO TO 11
      AB=ABS(C(L))
11    CONTINUE
      L=(I-1)*N
      DO 12 J=1,N
      L=L+1
12    C(L)=C(L)/AB
10    CONTINUE
13    CALL TRANS(A1,B1,C1,C,TI,N,M,K,1)
      CALL MCPY(C,T,N,N,0)
      RETURN
      END
```

```
      SUBROUTINE COMF(A,R,N,IND,C,T)
      DIMENSION A(1),C(1),R(1),T(1)
      IT=1
      CALL MCPY(R,C,1,N,0)
      IF(IND.EQ.1HO)GO TO 2
      CALL MCPY(R,T,1,N,0)
      DO 1 I=2,N
      IT=IT+N
      CALL GMPRD(C,A,T(IT),1,N,N)
      CALL MCPY(T(IT),C,1,N,0)
1     CONTINUE
      CALL GMTRA(T,C,N,N)
      RETURN
2     IT=(N-1)*N+1
      CALL MCPY(R,T(IT),1,N,0)
      DO 3 I=2,N
      IT=IT-N
      CALL GMPRD(A,C,T(IT),N,N,1)
      CALL MCPY(T(IT),C,1,N,0)
100   FORMAT(10I3)
3     CONTINUE
      CALL MCPY(T,C,N,N,0)
      RETURN
      END
```

```
      SUBROUTINE EGT(A,RR,RI,N,P)
      DIMENSION A(1),RR(1),RI(1),P(1)
      CALL SCLA(P,0.,N,N,0)
      EPS=0.01
      L = 0
      IS = 0
      DO 2 J = 1,N
      IF(ABS(RI(J)).LE.EPS) RI(J)=0.
      IF(IS.EQ.0) GO TO 4
      IS = 0
      L = L+N
      GO TO 2
4     L = L+1
      P(L) = 1.
      P(L+N) = 0
      DO 3 I=2,N
      L=L+1
      P(L)=P(L-1)*RR(J)-A(I-1)
      IF(J.EQ.1) GO TO 5
      AA=ABS(RR(J)-RR(J-1))
      IF(AA.LE.EPS.AND.RI(J-1).EQ.0..AND.RI(J).EQ.0.)
     1 P(L)=P(L)+P(L-N-1)
5     IF(RI(J).EQ.0.) GO TO 3
      P(L)=P(L)-RI(J)*P(L+N-1)
      P(L+N)=RR(J)*P(L+N-1)+RI(J)*P(L-1)
      IS = 1
3     CONTINUE
2     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE TRANS(A,B,C,T,TI,N,M,JP,IND)
      DIMENSION A(1),B(1),C(1),T(1),TI(1)
      DIMENSION L1(10),L2(10),P1(100),P2(100)
C       TI=A*T**A
C       TI=B  **B
C          C*T**C
C       IND = 0  T  AND  TI  AVAILABLE
C       IND = 1  ONLY  T  IS AVAILABLE
C       IND = 2  ONLY  TI IS AVAILABLE
      D=1.
      IF(IND.EQ.0) GO TO 3
      IF (IND.EQ.2) GO TO 2
      CALL MCPY(T,TI,N,N,0)
      CALL MINV(TI,N,D,L1,L2)
      GO TO 3
2     CALL MCPY(TI,T,N,N,0)
      CALL MINV(T,N,D,L1,L2)
3     IF(ABS(D).LE.0.01) CALL IOMT(D,1,1,2,15,'MAT. T SINGULAR')
      CALL GMPRD(TI,A,P1,N,N,N)
      CALL GMPRD(P1,T,A,N,N,N)
      CALL GMPRD(TI,B,P1,N,N,M)
      CALL MCPY(P1,B,N,M,0)
      CALL GMPRD(C,T,P1,JP,N,N)
      CALL MCPY(P1,C,JP,N,0)
      END
```

```
      SUBROUTINE CHEQ(A,N,F,B,C)
      DIMENSION A(1),B(1),C(1),F(1)
      NN=N*N
      M=N+1
      CALL MCPY(A,B,N,N,0)
      CALL SCLA(F,0.,1,N,0)
      DO 10 I=1,N
      IF=M-I
      DO 1 J=1,NN,M
1     F(IF)=F(IF)-B(J)
      F(IF)=F(IF)/I
      DO 2 J = 1,NN,M
2     B(J)=B(J)+F(IF)
      CALL GMPRD(A,B,C,N,N,N)
10    CALL MCPY(C,B,N,N,0)
      RETURN
      END
```

```
      SUBROUTINE RANK2(H,P,D,MT,K,IR,EPS)
      DIMENSION H(1),P(1),L1(20),M1(20),DP(20)
      CALL MCPY(H,P,MT,K,0)
      IST=0
100   FORMAT(5I5)
      CALL MFGR(P,MT,K,EPS,IR,L1,M1)
      CALL GTPRD(H,H,P,MT,K,K)
      DO 1 I=1,K
      IP=1+(I-1)*(K+1)
1     DP(I)=P(IP)
      IF(MT.EQ.K) CALL MCPY(H,P,MT,K,0)
      DD=DET(P,K)
      IF(K.GT.MT) DD=0.
      D=DD
      JR=IR
      IF(IR.NE.K) IR=0
      RETURN
      END


      SUBROUTINE JOIN(A11,A12,A21,A22,N1,M1,N2,M2,N,M,A)
      DIMENSION A11(1),A12(1),A21(1),A22(1),A(1)
      N=N1+N2
      M=M1+M2
      CALL RTIE(A11,A21,A,N1,M1,0,0,N2)
      IA=N*M1+1
      CALL RTIE(A12,A22,A(IA),N1,M2,0,0,N2)
      RETURN
      END
```

```
      FUNCTION DET(A,KC)
C       THIS FUNCTION FINDS THE DETERMINANT OF A MATRIX
C       USING DIAGONALIZATION PROCEDURE
      DIMENSION A(1),B(20,20)
      IREV=0
      DO 1 I=1,KC
      DO 1 J=1,KC
      IJ = I+(J-1)*KC
1     B(I,J)=A(IJ)
      DO 20 I=1,KC
      K=I
9     IF(B(K,I)) 10,11,10
11    K=K+1
      IF(K-KC) 9,9,51
10    IF(I-K) 12,14,51
12    DO 13 M=1,KC
      TEMP=B(I,M)
      B(I,M)=B(K,M)
13    B(K,M)=TEMP
      IREV=IREV+1
14    II=I+1
      IF(II.GT.KC) GO TO 20
      DO 17 M=II,KC
18    IF(B(M,I)) 19,17,19
19    TEMP=B(M,I)/B(I,I)
      DO 16 N=I,KC
16    B(M,N)=B(M,N)-B(I,N)*TEMP
17    CONTINUE
20    CONTINUE
      DET=1
      DO 2 I=1,KC
2     DET=DET*B(I,I)
      DET=(-1.)**IREV*DET
      RETURN
51    DET=0
      RETURN
      END


      SUBROUTINE DJOIN(A11,A12,A21,A22,N1,M1,N2,M2,N,M,A)
      DIMENSION A(1),A11(1),A12(1),A21(1),A22(1)
      IA=N*M1+1
      CALL RCUT(A,N1+1,A11,A21,N,M1,0)
      CALL RCUT(A(IA),N1+1,A12,A22,N,M2,0)
      RETURN
      END
```

```
      SUBROUTINE DREAL(A,B,N,M,DT,F,G,IND)
      DIMENSION A(1),B(1),F(1),G(1)
      COMMON M1(10),L1(10),P(200),D4(200),D2(200),D3(200)
     1,AA(100),D(100),D1(100),CC(100),AI(100)
      KND=0
      IM=100
      CALL MCPY(A,AA,N,N,0)
      IF(IND.EQ.'C') GO TO 2
      CALL AV(P,IM,DT,1,A1,A2)
      CALL CHT1(AA,P,N,IM,KND,D,D1,D2,CC,D3,D4,F)
      CALL MCPY(A,AI,N,N,0)
      CALL MINV(AI,N,D,L1,M1)
      CALL SCLA(AA,0.,N,N,0)
      CALL DCLA(AA,1.,N,0)
      CALL GMSUB(F,AA,AA,N,N)
      CALL GMPRD(AA,B,CC,N,N,M)
      CALL GMPRD(AI,CC,G,N,N,M)
      IF(IND.EQ.'D') RETURN
2     CONTINUE
      P(1)=0
      FF=-1.
      DO 1 I=2,IM
      FF=-FF
1     P(I)=FF/(I-1.)
      CALL SCLA(D,0.,N,N,0)
      IM=180
      CALL DCLA(D,1.,N,0)
      CALL GMSUB(AA,D,AA,N,N)
      CALL CHT1(AA,P,N,IM,KND,D,D1,D2,CC,D3,D4,F)
      CALL SMPY(F,1./DT,F,N,N,0)
      CALL MINV(AA,N,D,L1,M1)
      CALL GMPRD(F,B,D1,N,N,M)
      CALL GMPRD(AA,D1,G,N,N,M)
      RETURN
      END
```

```
      SUBROUTINE MTF(A,B,C,N,M,K,IND,F,W,Z)
      INTEGER H
      DIMENSION  A(1),B(1),C(1),F(1),W(1),Z(1)
      COMMON D(100),E(100),A0(100),B0(100),AH(100)
      CALL SMPY(B,-1.,B,N,M,0)
      CALL  SCLA(A0,0.,1,N,0)
      IF (IND)  16,17,17
17    CALL SCLA(Z,0.,N,K*M,0)
16    NN=N*N
      NNM=NN*M
      IA=1-N
      IW=1-N+NNM-NN
      IC=0
      DO 10 H=1,N
      IA=IA+N
      IB=1-N
      IW=IW+N-NNM
      IZ=1-N
      CALL MCPY(A,AH,N,N,0)
      CALL MCPY(A0,AH(IA),1,N,0)
      CALL CHEQ(AH,N,B0,D,E)
      IC=IC+K
      DO 10 J=1,M
      IB=IB+N
      IW=IW+NN
      CALL MCPY(A,AH,N,N,0)
      CALL MCPY(B(IB),AH(IA),1,N,0)
      CALL CHEQ(AH,N,F,D,E)
      CALL GMSUB(F,B0,F,1,N)
      IF(IND) 11,13,11
11    CALL MCPY(F,W(IW),1,N,0)
13    IC=IC-K
      IF(IND) 10,12,12
12    DO 14 I=1,K
      IC=IC+1
      IZ=IZ+N
      CALL SMPY(F,C(IC),D,1,N,0)
      CALL GMADD(Z(IZ),D,Z(IZ),1,N)
14    CONTINUE
10    CONTINUE
      CALL CHEQ(A,N,F,D,E)
      CALL SMPY(B,-1.,B,N,M,0)
      RETURN
      END
```

```
      SUBROUTINE CALYU($,B,R,AK,Z,U,V,N,M,K,NT,IO,IND)
      DIMENSION B(1),AK(1),R(1),U(1),V(1),Z(1)
      ISY=0
      IF(IO.EQ.2) GO TO 3
      IF(IND.EQ.'AN') CALL IFF($14,$16,26,' DO YOU WANT Y(T) = Y OR N')
3     CALL IFF($1,$4,26,' DO YOU WANT U(T) = Y OR N')
1     IF(IND.EQ.'RC') GO TO 11
      IF(IND.EQ.'PP') GO TO 12
      IF(IND.EQ.'PO') GO TO 13
      IF(IND.EQ.'AN') GO TO 15
      TYPE 101,IND
101   FORMAT(' INCORRECTLY SPECIFIED OPTION ',A5,'  FOR PLXU')
      RETURN
11    CALL GMPRD(B,R,V,N,M,M)
      CALL GMPRD(AK,V,Z,N,N,M)
      CALL GMPRD(U,Z,V,NT,N,M)
      CALL SMPY(V,-1.,U,NT,M,0)
      RETURN
12    CALL GMTRA(B,Z,M,N)
      CALL GMPRD(U,Z,V,NT,N,M)
      CALL MCPY(V,U,NT,M,0)
      RETURN
13    N=N/2
      CALL SMPY(B,-1.,Z,M,N,0)
      IB=N*M+1
      N2=N+N
      CALL MCPY(Z,B(IB),M,N,0)
      CALL GMTRA(B,Z,M,N2,0)
      CALL GMPRD(U,Z,V,NT,N2,M)
      CALL MCPY(V,U,NT,M,0)
      N=N2
      RETURN
16    ISY=1
14    CALL GMTRA(AK,Z,K,N)
      CALL GMPRD(U,Z,V,NT,N,K)
      CALL MCPY(V,U,NT,M,0)
      IF(ISY.EQ.1) GO TO 4
      RETURN
15    CALL GMTRA(R,Z,M,K)
      CALL GMPRD(U,Z,V,NT,K,M)
      CALL MCPY(V,U,NT,M,0)
      IO=0
2     RETURN
4     RETURN 1
      END
```

```
      SUBROUTINE SEL($,N,UC,K,N1,IUC)
      DIMENSION UC(1)
      COMMON AA(10),L1(10)
      IAL=0
      CALL IOMT(AA,1,N1,-1,29,'TYPE INDICES YOU WANT TO PLOT')
      IF(AA(1).LE..0) GO TO 40
      AN=N
      DO 15 I=1,N1
      IF(AA(I).GT.AN) IAL=1
      IF(AA(I).LE.0) GO TO 15
      L1(I)=AA(I)
      N2=I
15    CONTINUE
      N1=N2
      IF(IAL.EQ.0) GO TO 34
      DO 33 I=1,N
33    L1(I)=I
      N1=N
34    CONTINUE
      DO 16 I=1,N1
      I1=(L1(I)-1)*K+1
      I2=(I-1)*K+IUC
16    CALL MCPY(UC(I1),UC(I2),1,K,0)
      L1(1)=0
      RETURN
40    RETURN 1
      END
```

```
      SUBROUTINE MRLOC(N,C,E,DK,EV2V)
      DIMENSION C(1),E(1),DK(1),EV2V(1)
      N1=N
      CALL IFF(S11,S6,41,' DO YOU WANT LOCATIONS OF EIG.VAL. Y OR N')
11    CALL RLOC(N1,C,E,DK,EV2V)
      CALL IFF(S12,S6,21,' DO YOU WANT NEW PLOT')
110   FORMAT(' WHICH EIG.VAL. YOU WANT TO ELIMINATE '/
     1 ' TYPE LIM. VALUE')
12    TYPE 110
      CALL IOMT(C,1,N1,2,3,'R-R')
      CALL IOMT(A1,1,1,-2,6,'RR.LIM')
111   FORMAT(10I1)
      IF(A1.GE.0.) GO TO 12
      LL=0
      DO 10 I=1,N1
      IF(C(I).LE.A1) GO TO  10
      LL=LL+1
      C(LL)=C(I)
      E(LL)=E(I)
      N2=LL
10    CONTINUE
      N1=N2
      CALL IOMT(C,1,N1,2,7,'RED.R-R')
      CALL IOMT(E,1,N1,2,7,'RED.R-I')
      GO TO 11
6     CONTINUE
      RETURN
      END


      SUBROUTINE RLOC(N,RR,RI,X,Y)
      DIMENSION RR(1),RI(1),X(1),Y(1)
      X(1)=0.
      Y(1)=0.
      DO 1 I =1,N
      I1=2*I
      I2=I1+1
      X(I1)=0
      Y(I1)=0
      X(I2)=RR(I)
      Y(I2)=RI(I)
1     CONTINUE
      CALL INITT(30)
      CALL BINITT
      CALL SLIMX(200,800)
      CALL SLIMY(200,800)
      CALL NPTS(I2)
      CALL LINE(12)
      CALL STEPS(2)
      CALL SYMBL(6)
      CALL CHECK(X,Y)
      CALL DSPLAY(X,Y)
      CALL TINPUT(IW)
      RETURN
      END
```

```
      SUBROUTINE PLOTN(N,NP,X,Y)
      DIMENSION X(1),Y(1),LIN(6),Z(2)
      DATA LIN/12,23,34,45,56,67/
      ISCALE=0
6     CALL INITT(30)
      CALL BINITT
      CALL SLIMX(100*N,800)
      CALL NPTS(NP)
      CALL YFRM(2)
      CALL XFRM(4)
      DO 2 I=1,N
      J=(I-1)*NP+1
      CALL LINE(LIN(I))
      IF(I.GT.1.AND.ISCALE.EQ.0) CALL DINITY
      IF(ISCALE.EQ.1) CALL DLIMY(YMIN,YMAX)
      IF(I.GT.1) CALL YLOC(-100*(I-1))
      CALL CHECK(X(1),Y(J))
      IF(I.GT.1) CALL XLAB(0)
      CALL DSPLAY(X(1),Y(J))
2     CONTINUE
      CALL CURSE
      CALL TINPUT(IW)
      CALL IFF(S1,S3,48,
     1' DO YOU WANT NEW PLOT WITH COMMON SCALE = Y OR N')
1     CALL IOMT(Z,1,2,-1,9,'YMIN,YMAX')
      YMIN=Z(1)
      YMAX=Z(2)
      ISCALE=1
      GO TO 6
3     RETURN
      END


      SUBROUTINE CURSE
10    CALL OCURSR(ICHAR,IX,IY)
      CALL MOVABS(IX,IY)
      IF(ICHAR.EQ.127) RETURN
      CALL ANMODE
      CALL ANCHO(ICHAR)
      GO TO 10
      END
```

```
      SUBROUTINE RESP(A,DT,N,NT,X0,XADD,RES,B,M)
      DIMENSION A(1),X0(1),XADD(1),RES(1),B(1)
     1,T(300)
      COMMON L1(10),M1(10),D(100),C(100),P1(100)
      EQUIVALENCE (T(1),D(1))
      COMMON /A1/H(100)
C     XADD = NT*N  ;  RES = N*NT
      IPL=0
      CALL SCLA(XADD,0.,N,NT,0)
      CALL SCLA(H,0.,1,N,0)
      CALL MCPY(A,C,N,N,0)
      CALL MINV(C,N,DD,L1,M1)
      IF(ABS(DD).LE..001)       CALL IOMT(DD,1,1,2,2,'DD')
      CALL SCLA(D,0.,1,N,0)
      IF(M.LT.1) GO TO 3
      L=N
      F=1.
      DO 8 I=2,M
      F=F*(I-1.)
      DO 8 J=1,N
      L=L+1
8     B(L)=B(L)*F
      DO 2 I=1,M
      J=M-I+1
      IB=(J-1)*N+1
      IF(IPL.NE.0)       TYPE 100,I,J,IB
100   FORMAT(10I3)
      CALL GMADD(B(IB),D,D,1,N)
      CALL GMPRD(C,D,H(IB),N,N,1)
2     CALL MCPY(H(IB),D,1,N,0)
3     CALL GMADD(X0,H,X0,1,N)
      IF(IPL.NE.0)       CALL IOMT(X0,1,N,3,3,'XON')
      IF(IPL.NE.0) CALL IOMT(A,N,N,3,1,'A')
      CALL OUT2(A,DT,N,NT,X0,XADD,RES)
      IF(L1(1).NE.11) GO TO 10
      TYPE 101
101   FORMAT(' MATRIX NON CYCLIC')
      IND=3
      CALL OUT(A,DT,N,NT,X0,IND,XADD,RES)
      TYPE 100,IND,L1(1)
10    CONTINUE
      CALL SCLA(XADD,0.,N,NT,0)
      IF(IPL.NE.0)       CALL IOMT(RES,N,6,3,3,'RES')
      IF(IPL.NE.0)       CALL IOMT(H,N,M,3,1,'H')
      IF(M.LT.1) GO TO 1
      X=-DT
      L=0
      DO 4 I=1,NT
      X=X+DT
      DO 4 J=1,M
      L=L+1
      IF (J.GT.1) GO TO 5
      T(L)=1.
      F=1.
      GO TO 4
5     F=F*(J-1.)
      T(L)=X**(J-1)/F
4     CONTINUE
      CALL GMPRD(H,T,XADD,N,M,NT)
      IF(IPL.NE.0)       CALL IOMT(XADD,N,6,2,4,'XADD')
      IF(IPL.NE.0)       CALL IOMT(T,M,6,3,1,'T')
1     CALL GMSUB(RES,XADD,RES,N,NT)
      CALL GMTRA(RES,XADD,N,NT)
      IF(IPL.NE.0)       CALL IOMT(RES,N,6,3,3,'RES')
      RETURN
      END
```

```
      SUBROUTINE OUT2(A,DT,N,NT,X0,T,P1)
      DIMENSION A(1),X0(1),T(1),P1(1)
      COMMON L1(10),M1(10),Q(100),P2(100),TI(100),A1(100)
100   FORMAT(10I3)
      CALL MCPY(A,A1,N,N,0)
      EPC=10.**(-7)
      EPS=0.01
      DO 4 I=1,N
      KK=2*I+1
      DO 2 J=1,N
      P1(J)=RAN(KK)
      IF(I.EQ.1) P1(J)=1.
2     CONTINUE
      CALL COMF(A1,P1,N,1HO,T,TI)
      D=DET(T,N)
      IF(ABS(D).GT.EPS) GO TO 3
      CALL IOMT(P1,1,N,3,6,'VEC-TC')
4     CONTINUE
      CALL IOMT(D,1,1,2,6,'DET-TC')
3     CONTINUE
      IF(ABS(D).GT.EPC) GO TO 10
      TYPE 101
101   FORMAT(' MATRIX NON CYCLIC')
      L1(1)=11
      RETURN
10    CONTINUE
      CALL CHEQ(A,N,Q,P1,P2)
      Q(N+1)=1
      DO 1 I=1,N
      IC=N-I+1
1     P1(IC)=-Q(I)
      IHSBG=1
      IF(IHSBG.EQ.0) GO TO 5
      CALL MCPY(A,Q,N,N,0)
      CALL HSBG(N,Q,N)
      CALL ATEIG(N,Q,P2,TI,L1,N)
      CALL ORD(P2,TI,N)
      GO TO 6
5     CALL POLRT(Q,P1,N,P2,TI,IE)
      IF(IE.NE.0) TYPE 100,IE
6     CALL EGT(P1,P2,TI,N,Q)
      CALL GMPRD(T,Q,P1,N,N,N)
      CALL MCPY(P1,A1,N,N,0)
      CALL OUTN(A1,DT,P2,TI,N,NT,X0,P1)
      CALL GMTRA(P1,T,N,NT)
      L1(1)=0
      RETURN
      END
```

```
      SUBROUTINE OUT(A,DT,N,K,X0,IND,P,X)
      DIMENSION A(1),X0(1),P(1),X(1)
      DIMENSION L1(1),D(100),D3(200),D4(200),D1(200),D2(100)
     1,AF(100),FM(100),C(100),P1(200)
      CALL MCPY(A,AF,N,N,0)
      IA=N*N+1
      A(IA)=0.
      T=0.
      IM=180
      JND=0
      IF(IND.LT.2) GO TO 4
      IND=IND-2
      JND=1
4     CONTINUE
      CALL MCPY(X0,X,1,N,0)
      CALL SCLA(C,0.,C,N,N,0)
      CALL DCLA(C,1.,N,0)
      IX = 1
      DO 1 I = 2,K
      IX = IX+N
      IF(IND.EQ.1.AND.I.NE.2) GO TO 2
      T = T+DT
      CALL AV(P1,IM,T,1,A1,A2)
      IF(JND.EQ.0) GO TO 5
      KND=0
      CALL CHT1(AF,P1,N,IM,KND,D,D1,D2,C,D3,D4,FM)
      A(IA)=KND
      GO TO 6
5     CONTINUE
      CALL CHT(AF,P1,N,IM,D,D1,D2)
      CALL MATPL(AF,N,1,D,D1,D2,FM)
6     CONTINUE
      CALL MCPY(FM,C,N,N,0)
      GO TO 3
2     CONTINUE
      CALL GMPRD(FM,C,D,N,N,N)
      CALL MCPY(D,C,N,N,0)
3     CONTINUE
      CALL GMPRD(C,X0,X(IX),N,N,1)
1     CONTINUE
      CALL GMTRA(X,P,N,K)
      RETURN
      END
```

```
      SUBROUTINE OUTN(Q,DT,ALV,RI,N,NT,X0,X)
      DIMENSION Q(1),ALV(1),RI(1),X0(1),X(1)
      DIMENSION L1(10),M1(10),P(100)
      IST1=0
      IF(IST1.NE.0) CALL IOMT(Q,N,N,3,3,'T-M')
      IF(IST1.NE.0) CALL IOMT(ALV,1,N,2,3,'ALV')
      EPS=0.01
      RI(N+1)=1000.
      ALV(N+1)=0
      ALV(N+1)=1000.*ALV(N)+11.11
      IST=0
      NM=N
      CALL MCPY(Q,P,N,N,0)
      CALL MINV(P,N,D,L1,M1)
102   FORMAT(5I3,4E10.3)
      IF(ABS(D).LE.EPS) CALL IOMT(D,1,1,2,5,'DET.T')
      CALL GMPRD(P,X0,X,N,N,1)
      CALL MCPY(X,X0,1,N,0)
      T=-DT
      DO 1 I=1,NT
      T=T+DT
      II1=1
      L=0
7     CONTINUE
      I1=II1
      NM=0
      DO 5 J=I1,N
      IF(ABS(RI(J)).LE.EPS) GO TO 8
      AK=SQRT(X0(J)**2+X0(J+1)**2)
      E=EXP(ALV(J)*T)
      TET=ATAN2(X0(J+1),X0(J))
      OMT=-ABS(RI(J))*T
      P(L+1)=AK*E*COS(OMT+TET)
      P(L+2)=AK*E*SIN(OMT+TET)
      L=L+2
      IF(L.GE.N) GO TO 21
      II1=I1+2
      IF(IST.EQ.1) TYPE 102,L,I1,J,J,J,E,AK,OMT,TET
      GO TO 7
8     CONTINUE
      NM=NM+1
      DAL=ALV(J)-ALV(J+1)
      IF(IST1.EQ.0) GO TO 10
      CALL IOMT(DAL,1,1,2,3,'DAL')
      CALL IOMT(ALV,1,N,1,4,'ALVV')
      TYPE 102,J
10    CONTINUE
      DRI=ABS(RI(J)-RI(J+1))
      IF(IST1.NE.0) TYPE 102,J,J,J
      IF(ABS(DAL).GT.EPS.OR.DRI.GT.EPS) GO TO 6
      IF(IST1.NE.0) TYPE 102,NM,J
5     CONTINUE
6     II1=J+1
      AL=ALV(J)
      L=L+1
       IF(IST.EQ.1) TYPE 102,NM,J,I1,L
      L=L-1
      E=EXP(AL*T)
      DO 2 J=1,NM
      L=L+1
      Y=0.
      F=1.
      TKT=1.
      DO 3 K=J,NM
      KT=K-J
      KI=KT+1
      IF(KT.EQ.0) GO TO 4
      TKT=TKT*T
      F=F/KT
4     CONTINUE
      Y=Y+TKT*F*E*X0(K+L-J)
      IF(IST.EQ.1) TYPE 102,I,J,K,KT,L,AL,T,Y,E
3     CONTINUE
      P(L)=Y
2     CONTINUE
      IF(II1.LE.N) GO TO 7
21    NM=N
      IX=(I-1)*NM+1
      CALL GMPRD(Q,P,X(IX),N,N,1)
      IF(IST.EQ.1) CALL IOMT(P,1,NM,2,1,'Y')
1     CONTINUE
      RETURN
      END

      SUBROUTINE ORD(RR,RI,N)
      DIMENSION RR(1),RI(1)
      N1=N-1
      DO 1 I=1,N1
      J1=I+1
      DO 2 J=J1,N
      IF(RR(I).GE.RR(J)) GO TO 2
      TT=RR(I)
      RR(I)=RR(J)
      RR(J)=TT
      TT=RI(I)
      RI(I)=RI(J)
      RI(J)=TT
      IF(ITY.EQ.0) GO TO 2
      TYPE 100,I,J
100   FORMAT(' I= ',I3,' J= ',I3)
      CALL IOMT(RR,1,N,3,2,'RR')
      CALL IOMT(RI,1,N,3,2,'RI')
2     CONTINUE
1     CONTINUE
      I=0
3     I=I+1
      IF(I.GT.N) RETURN
      IF(ABS(RI(I)).LE.0.00001) GO TO 3
      IF(RI(I).GT.0.) GO TO 4
      RI(I)=-RI(I)
      RI(I+1)=-RI(I+1)
4     I=I+1
      GO TO 3
      END
```

```
      SUBROUTINE CHT1(A,P,N,M,IND,C,B,D,E,Q,R,Y)
      DIMENSION A(1),P(1),C(1),B(1),D(1),E(1),Q(1),R(1),Y(1)
      KM=2
      KM=1
      K=(M-1)/N+1
      L=(K-1)/N+1
      IS=(L-1)/N+1
      JND=1
      IF(IND.NE.0) GO TO 1
      IF(K.GT.KM) JND=2
      IF(L.GT.KM) JND=3
      IF(IS.GT.KM)JND=4
      JND=JND
1     CONTINUE
      IS=0
      IF(IS.NE.0) TYPE 100,N,M,IND,K,L,IS,JND
100   FORMAT (10X,7I5)
      NN=N*N
      IF (IND.EQ.1) GO TO 11
      CALL FFC(A,N,C,B,D)
      IF (IND.EQ.2) GO TO 12
      CALL FFC(C,N,D,B,Q)
      IF (IND.EQ.3) GO TO 13
      CALL FFC(D,N,E,B,Q)
      CALL FFD(P,E,R,N,IS,Y,Q)
      IS=N
23    CONTINUE
      CALL FFB(R,D,Q,N,IS,N,B)
      IS=N
22    CONTINUE
      CALL FFB(Q,C,B,N,IS,1,R)
      IS=N
21    CONTINUE
      CALL FFA(B,A,Y,N,IS,C,D)
      RETURN
11    IS=M
      CALL SCLA(B,0.,N,IS,0)
      CALL MCPY(P,B,1,IS,0)
      GO TO 21
12    IS=K
      CALL SCLA(Q,0.,N,IS,0)
      CALL MCPY(P,Q,N,IS,0)
      GO TO 22
13    IS=L
      CALL SCLA(R,0.,NN,IS,0)
      CALL MCPY(P,R,NN,IS,0)
      GO TO 23
      RETURN
      END
```

```
      SUBROUTINE CHT(A,P,N,M,B,C,D)
      DIMENSION A(1),P(1),B(1),C(1),D(1)
      K=(M-1)/N+1
      N1=N+1
      NN=N*N
      CALL CHEQ(A,N,B,C,D)
      CALL SMPY(C,0.,C,N,N,0.)
      J1=1-N
      DO 10 I=1,N
      J1=J1+N
      DO 10 J=J1,NN,N1
10    C(J)=B(I)
      DO 20 I=2,N
      I1=I-1
      DO 20 J=1,I1
      J1=1+N-I+J
      IL=I-N
      JL=J-N
      DO 20 L=1,N
      IL=IL+N
      JL=JL+N
20    C(IL)=C(IL)-B(J1)*C(JL)
      L=2*(K/2*2-K)+1
      I1=N*(K-1)+1
      FF=-L
      CALL SMPY(P(I1),FF,B,1,N,0)
      DO 30 I=2,K
      CALL GMPRD(B,C,D,1,N,N)
      I1=I1-N
      IF(L) 31,31,32
31    CALL GMSUB(D,P(I1),B,1,N)
      GO TO 30
32    CALL GMADD(D,P(I1),B,1,N)
30    L=-L
      RETURN
      END
```

```
      SUBROUTINE MATPL(A,N,IND,P,B,D,FM)
      DIMENSION A(1),P(1),B(1),D(1),FM(1)
      IF(IND-1) 2,2,3
2     CONTINUE
      CALL SMPY(FM,0.,FM,N,N,0)
      CALL MCPY(FM,B,N,N,0)
      CALL DCLA(B,1.,N,0)
3     CONTINUE
      DO 1 J=1,N
      CALL SMPY(B,P(J),D,N,N,0)
      CALL GMADD(FM,D,FM,N,N)
      CALL GMPRD(B,A,D,N,N,N)
      CALL MCPY(D,B,N,N,0)
1     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FFA(P,A,Y,N,IS,C,D)
      DIMENSION A(1),C(1),D(1),P(1),Y(1)
      CALL SCLA(Y,0.,N,N,0)
      CALL DCLA(Y,P(IS),N,0)
      DO 1 I=2,IS
      J=IS-I+1
      CALL GMPRD(Y,A,C,N,N,N)
      CALL SCLA(D,0.,N,N,0)
      CALL DCLA(D,P(J),N,0)
      CALL GMADD(D,C,Y,N,N)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE FFB(P,C,B,N,IS,K,D)
      DIMENSION P(1),C(1),B(1),D(1)
      L=N*K
      IP=(IS-1)*L+1
      CALL MCPY(P(IP),B,K,N,0)
      DO 1 I=2,IS
      J=IS-I+1
      CALL GMPRD(B,C,D,K,N,N)
      IP=(J-1)*L+1
      CALL GMADD(P(IP),D,B,K,N)
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE AV(A,N,DT,J,A1,AN)
      DIMENSION A(1)
      IF(J-1) 12,11,12
   11 A(1)=1.
      GO TO 13
   12 A(1) = A1*DT/((J-1.)*N)
   13 CONTINUE
      DO 1 I = 2,N
      B = I-1.+(J-1)*N
    1 A(I)=A(I-1)*DT/B
      AN=A(N)
      IF(J/2*2-J)21,20,21
   20 CALL SMPY(A,-1.,A,1,N,0)
   21 RETURN
      END
```

```
      SUBROUTINE FFC(A,N,C,B,D)
      DIMENSION A(1) ,B(1),C(1),D(1)
      N1=N+1
      NN=N*N
      CALL CHEQ(A,N,B,C,D)
      CALL SMPY(C,0.,C,N,N,0.)
      J1=1-N
      DO 10 I=1,N
      J1=J1+N
      DO 10 J=J1,NN,N1
   10 C(J)=B(I)
      DO 20 I=2,N
      I1=I-1
      DO 20 J=1,I1
      J1=1+N-I+J
      IL=I-N
      JL=J-N
      DO 20 L=1,N
      IL=IL+N
      JL=JL+N
   20 C(IL)=C(IL)-B(J1)*C(JL)
      CALL SMPY(C,-1.,C,N,N,0)
      RETURN
      END
```

```
      SUBROUTINE FFD(P,E,R,N,IS,C,Q)
      DIMENSION P(1),E(1),R(1),C(1),Q(1)
      NN=N*N
      CALL MCPY(E,Q,N,N,0)
      IF(IS.LT.3) GO TO 2
      DO 1 I=3,IS
      I1=(I-3)*NN+1
      I2=I1+NN
  100 FORMAT(10I4)
    1 CALL GMPRD(E,Q(I1),Q(I2),N,N,N)
    2 CONTINUE
      DO 3 L=1,N
      IR=(L-1)*NN+1
      CALL MCPY(P(IR),R(IR),N,N,0)
      DO 3 J=2,IS
      DO 3 I =1,N
      IP=(J-1)*N+I
      IQ=I+(L-1)*N+(J-2)*NN
      JP=(IP-1)*NN+1
      CALL SMPY(P(JP),Q(IQ),C,N,N,0)
    3 CALL GMADD(R(IR),C,R(IR),N,N)
      RETURN
      END
```